A METHOD FOR THE NUMERICAL SIMULATION OF A PHYSICAL PHENOMENON WITH A PREFERENTIAL DIRECTION

The invention concerns a method of simulating behavior of a flow interacting with an object, the method providing a simulated numerical representation in N dimensions, $N \geq 3$, that is composed of a plurality of approximated values at a multitude of points in at least a part of space where the flow interacts with the object, the approximated values being of a physical parameter u of the flow to which is associated a velocity field $\vec{a}$ which determines a preferential direction, by means of a numerical scheme wherein at least one spatial p^{th} derivative D_p, $p \geq 1$, of the physical parameter u is approximated at the points of the part of space.

The numerical simulation of physical parameters related to a velocity field is instrumental in many practical applications for the conception, optimization and dimensioning of various appliances and commercial products, related to internal or external flow. Numerical simulations have become a valuable alternative to or complement of physical experiments such as performed in e.g. a wind tunnel or a towing tank.

As an example, numerical codes in the field of fluid dynamics can compute the flow for the conception of aircraft, cars, ships, turbines, ventilation and so on. Numerical simulations can also be used for the modeling of multi-phase flow as encountered in oil recovery or in combustion ; or in various electro-magnetic applications ; or multi-physics flow like magneto-hydrodynamics ; or in meteorological phenomena like weather prediction ; or biological flow in arteries, the heart or the lungs ; or forest fire simulation ; or shallow water simulation for river and channel flow and so on.

It should be noted in this regard that for certain applications, the numerical simulation is the only feasible way to obtain practical information about the behavior of objects interacting with flow. These applications are characterized by extreme difficulties in, or the impossibility of performing physical measurements, or the practical use of them. This is in particular the case in weather prediction, hypersonic flow or combustion. Numerical simulations are in any case a valuable complement to traditional measurements when the measurements are available. Numerical simulations are currently widely used to reduce the cost and time of the design-cycle of commercial products.

Background of the Invention

The numerical simulation of physical phenomena is well known, and numerical solutions obtained from computer codes are used in many domains of industry. In the following example, we will recapitulate the steps which enter into the daily routine of the application of numerical simulations. The domain of application of the example is Computational Fluid Dynamics (CFD), but the methodology applies equally to other domains and applications. Consider a company which makes aircraft, and who wishes to design or to optimize a wing for the lift to drag ratio. The design department decides to use the tool of numerical simulations since wind tunnel tests are expensive. Their CFD department responds to this request in the following way.

1. A decision is made on which part of physics is important to describe in enough detail the phenomena in a meaningful simulation. In many fluid dynamics cases, the basic fluid dynamics equations like the Navier-Stokes equations are adequate. Perhaps a more simple description can be used. If viscous effects are less important, the Euler equations can be used. At low Mach number, an incompressible approximation can be used. Maybe the potential flow approximation is sufficient. On the other hand, a turbulence model may be needed, together with acoustic models, and a description of the deflection of the wing under aerodynamic forces.

The physical description is expressed in a mathematical model comprising of one or more equations which describe the relations between the physically relevant parameters (mass, density, energy, forces, impulse, pressure, volume, temperature, species concentration, charge density, magnetic field, ...). In general, the mathematical model uses spatial derivatives of one or more of the parameters, as well as time derivatives. Together with the number of space dimensions, the physical model determines the number of variables which are needed in the solution. In the case of the Navier-Stokes equations in three dimensions, five variables which depend on space and time describe the physical solution. These five variables can be e.g. the density, the momentum in the three directions and the energy. This set of variables is commonly known as conservative variables. Other combinations are possible, such as the density, the three velocity components and the pressure, which is known as the set of primitive variables. Yet another set of variables is known as entropy variables, since one of the variables is chosen to be proportional to the entropy. A transformation is always possible between different sets of independent variables.

2. The next step is to choose a computer program which can compute a solution to the problem which is compatible with the physical description of the problem. The simulation code will be run on a computer. While the physics and the mathematical

model describe a continuum, the computer can only work with discrete values in a limited number of points. The solution of the mathematical model with a computer program therefore involves the procedure of discretization, which consists of two steps.

First, the continuous variable is replaced by a discrete variable in space. A grid defines points in space where the values of the parameters are stored.

The second step is to rewrite the relations between the continuous variables of the mathematical model in relations between discrete variables. In particular, the derivatives which appear in the mathematical model are replaced by their discrete equivalent, which are differences which are locally calculated using the values of the discrete variables of the surrounding grid points. This step involves an approximation and is not exact.

In general, a big enough company has the means and experience to develop and maintain an in-house computer code. Such a code is tuned to the specific needs of the company. It may compute solutions to only a very restricted number of physical problems, but it is likely to be up to date. The developments in the domain are followed, and e.g. new space discretizations, faster solution techniques such as accelerators and so on are coded. After testing, the code is ready for production runs. Smaller enterprises often can afford only a small CFD group, which uses programs from a specialized provider. This is normally a more general code, since it will have to accommodate a larger spectrum of clients with different applications.

3. The computation of the numerical solution is preceded by a pre-processing step. A geometrical description of the wing is used to generate a computational grid. This grid defines the points in space where the solution is calculated. Together with this grid an initial solution is generated. In the case of the Navier-Stokes equations, the five independent variables are given a certain value for each of the grid points, corresponding to e.g. uniform flow at the design Mach number. The grid and the initial solution are written to one or more files.

4. The next step is to run the simulation code. The program performs the following actions:

a) The program reads the grid, and stores for each point the coordinates in the memory of the computer.

b) It reads the initial solution and stores for each grid point the value of the physical parameters in memory.

c) Additional files are read which specify computational parameters such as the number of iterations or a convergence criterion, which physical model to use, which of the coded space discretizations to use, and additional information if necessary.

c) Boundary conditions are specified which are used at the extremities of the part of space under consideration. They represent inflow and outflow from the domain, the way the wing influences the flow, and any other interaction of the flow with its surroundings.

d) Using the grid, the initial solution and the boundary conditions, it solves iteratively and approximately the discrete mathematical equations which describe the physical problem.

e) The code stops after a specified number of iterations, or when a solution is obtained which is accurate enough.

f) The code writes the solution to a file. This output solution consists of the value of physical variables in each of the grid points, where the number of physical variables corresponds to the problem. Another output file may be written which contains the grid.

5. A post-processing program is used to read the grid file and the output file of step 4f. Normally this very general program permits to extract any desired physical parameter which can be computed from the solution. If the solution is written in the form of conservative variables, this program can compute the pressure, which is needed for the lift and drag. It can also compute variables like the Mach number, the temperature and so on.

6. The results are used to modify or improve the geometry.

Steps 3-6 are repeated until the design is optimal. In the case of insufficiency of the physical model, a more elaborate one is chosen, and coded if necessary. If there is insufficient resolution of the simulated physical phenomena on parts of the grid, a finer grid is generated.

The above describes the typical computation of a numerical solution and its use in an industrial environment. A similar procedure is followed when the purpose is to design or to optimize part of a car, a ship, a vehicle in general, a turbo machine, a ventilation system, a wind mill, a pump, a combustion engine, a channel, the mixture of steam and oil or water and oil in porous media for oil recovery, and so on, or to describe the evolution of flow such as in weather prediction, climate modeling or in a forest fire. In step 1, the physical model depends on the problem under consideration. In a magneto-hydrodynamics application, the description of the electro-magnetic field enters. This entails the use of additional variables in the solution. In an application with different chemical species, the concentration of each of the species enters, together with a description of the transformation between the different chemical compounds. In

multi-phase flow, such as e.g. oil and steam, or water and air, all fluids require equations and variables for a proper description.

The pre-processing and post-processing, steps 3 and 5 respectively, are normally separate programs. Occasionally, they can be incorporated in the numerical solver.

In step 3 the grid is generated once for a given geometry. In current practice a grid can contain many millions of points. Very large computational grids are normally composed of different blocks. Such a multi-block grid can then be distributed to multiple processors or even to different computers for a parallel computation. Some numerical simulation programs are able to refine or derefine automatically the grid locally using an error estimation based on the computed solution. Some programs use such a capability to compute a time-accurate simulation where a concentration of grid points follows a physical phenomena like e.g. a vortex or a shock. In the case of a time-accurate simulation, step 4f is repeated for each intermediate solution of interest. This is e.g. the case for the instationary behavior of a turbo machine where the radiated sound is computed. Another example is combustion.

The practical use of numerical simulations is illustrated in step 5, where the output of the program is transformed into a useful result. The output of the numerical simulation can serve many purposes, depending on the application. Quite often, this involves the pressure, since this relates to forces on the object under design. For a wing, this can be the Mach number for restricting the shock strength. For turbo machines, this can be the pressure on the blades, or the thermal load. For a heart valve, this can be the velocity field which needs to be smooth to avoid cluttering. For a forest fire simulation, this is again the velocity field to predict the likely progress of the fire.

Normally, the person who runs the program makes use of the output as illustrated in step 6. The output can also be fed back without human interference into the program. This is e.g. the case for inverse methods, where the desired pressure distribution is prescribed in the form of a goal. The geometry of the object and the related grid possess degrees of freedom which allow optimization to reach the goal. Remark that the purpose in the example above is the optimal geometry of the wing. The numerical solution is a convenient tool to achieve this goal.

At the core of the numerical simulation is the approximation of space derivatives as used in step 4d. The discrete mathematical equations which are used in that step are the subject of the invention.

For the computation of each value of a space derivative at a grid point, also called node, one of the methods, known under the name of Finite Differences, consists of the computation of an approximate value of each space derivative as a linear combination of values of the parameter(s) taken at each of the points of a set of grid points, called

stencil, which are in general points surrounding the point of computation, and with weighting coefficients chosen in such a way that the best possible approximation results.

Another method is the Finite Volume method, which is very closely related to Finite Differences, and is based on an integral formulation using fluxes through a control volume. These fluxes are in turn a function of a number of the nodal unknowns, defining again a stencil. The number and relative importance of nodes involved in the fluxes allows for optimizing the approximation according to predefined criteria.

Yet another method is the Finite Element method, where the space is subdivided into elements, containing nodes at which the approximation of the physical parameter(s) u is stored. Basis functions ϕ, also called interpolation functions, are defined on the elements, which are used to approximate the physical value u on the element. The integral of the derivative is computed on the element, using a test function ψ, also called weighting function. The combination of ϕ and ψ is then chosen in such a way that the approximation of the derivative is optimal.

Then, there are Distribution Methods. They share with Finite Elements the representation of the unknown(s) u on the element. One variant, the Residual Distribution method shares with the Finite Volume method a volume over which the integral of the derivative is computed. The integral of the derivative is split into parts which are distributed over the nodes. Another variant, the Flux Vector Distribution method uses like the Finite Volume method fluxes through a control volume. These fluxes are distributed over the nodes. In both cases, the characteristics and the optimization of the discretization follow from the choice of the distribution coefficients.

All these methods have in common that the approximation is ultimately dependent on the values of the parameter(s) at the nodes of the stencil, and that certain discretization parameters, which are inherent to the particular numerical framework under consideration, influence the accuracy of the approximation and the numerical properties of the numerical simulation of the physical phenomena.

The choice of the stencil, the values of the discretization parameters and the associated numerical method are commonly called the numerical scheme of the numerical simulation of physical phenomena.

Stencils are commonly classified according to the relative position of the points with respect to the advection direction in velocity fields. In an upwind discretization, the stencil has nodes which are upstream of the point where the derivative is computed. Another type of discretization is central, where the stencil is symmetric with respect to the advection speed for the point where the derivative is computed. Finally, there are many discretizations which are a combination of the two.

Concerning the grid, one can use a regular or structured mesh, formed by N families

of rectilinear or curved lines, where the mesh points are located at the intersections. The number of families of lines corresponds to the number of space dimensions. Another type of grid, called unstructured grid, uses a connection table to identify the points of a stencil, and the points are in general less regularly positioned in space. However, unstructured grids can be locally perfectly regular, allowing to discern locally families of lines.

The accuracy of the simulation depends directly on the grid spacing. If the error in each space derivative reduces linearly with the mesh size, the scheme is said to be of first order. If the error reduces quadratically with the mesh size, the scheme is said to be of second order. In general, when the error reduces with the power M of the mesh size, the scheme is said to be of order M.

For a given application, the choice of the scheme is linked with practical constraints of the computation, called numerical constraints, which correspond to certain priorities imposed by the user on resulting errors of the simulation. Among the numerical constraints, one may refer to consistency, convergence and stability of the simulation, the desired order of the scheme (industrial applications require an order of at least 2), the diffusion and dispersion properties of the numerical scheme, and computational cost (which may impose a maximum size of the stencil, generally indicated by compactness).

The order of the approximation and the extent of the stencil are related. On a stencil of given extent, the order of accuracy of an approximation is limited. In other words, for an approximation of a given order, the stencil has to have a certain minimum extent. It depends on the numerical constraints of the application if the size of the stencil or the order of the approximation is predominant. When the appropriate combination of the size of the stencil and the desired accuracy is chosen, remaining degrees of freedom can be used for optimizing other numerical constraints.

An extended description of the above, and of numerical methods in general can be found in (*D1*), "Numerical computation of internal and external flow", C. Hirsch, J. Wiley & Sons, New York, Vol. 1, Fundamentals of numerical discretization, 1988 and Vol. 2, Computational methods for inviscid and viscous flows, 1990.

There are various methods to obtain the discretization parameters appearing in the specific numerical framework. In general, a truncated Taylor series expansion with respect to the computational point is used for the value of the physical parameter(s). Given that only the first terms of the Taylor series are used, the value obtained is accompanied with an error, called truncation error, which depends in particular on the degree of the last retained term in the series. The discretization parameters are then chosen such that they optimize the error in the discretization of the derivative, which is a function of the values resulting from the truncated Taylor series. Other criteria

follow from considering the Fourier transform of the discretization, thereby choosing the discretization parameters in such a way that the behavior of certain Fourier components is optimized. This means tuning the dissipation and dispersion of the numerical scheme, optimizing for the stability of the numerical scheme, or for the damping of certain components resulting in accelerated convergence. Other optimizations related to the choice of the discretization involve computational cost, ease of use in multi-block codes, ease of coding and others.

As pointed out, for each application, the most appropriate choice has to be made on :

- the numerical formulation with associated discretization parameters which is used for the simulation,
- the stencil itself, and notably the number of points and their relative position with respect to the point of computation,
- the optimization of the discretization, e.g. based on the truncation error in the truncated Taylor series expansion used in the computation of the discretization parameters, or based on the behavior of certain Fourier components, or other.

Many methods for numerical simulation have been proposed and are used in industry with more or less satisfactory performances. The above mentioned books *D1* give an overview of the mathematical models, discretization techniques, numerical schemes and solution methods for the numerical simulation of space derivatives in flow phenomena. In particular, in *D1* Volume 1, part 2, chapter IV, pages 167-180, the fundamentals are described of the method of Finite Differences, on a regular grid in one dimension. As explained, the weighting coefficients used in the linear combination can be determined if the number of points of the stencil is coherent with the order of the desired discretization. In particular, the book mentions the general method of Hildebrand for determining Finite Difference formulas.

An example is a general one-dimensional discretization for a derivative at node i which uses the stencil between the nodes $i-m$ and $i+n$, where m and n are given positive integers. On the stencil $S=(i-m, i-m+1, \ldots, i-1, i, i+1, \ldots, i+n-1, i+n)$, the approximation of the first derivative can be written as

$$\begin{aligned} u_x = \frac{1}{\Delta x}(\quad & a_{-m}u_{i-m} + a_{-m+1}u_{i-m+1} + \ldots \\ & + a_{-1}u_{i-1} + a_0 u_i + a_1 u_{i+1} + \ldots \\ & + a_{n-1}u_{i+n-1} + a_n u_{i+n}) \, . \end{aligned} \tag{1}$$

The coefficients a_j, $i-j \in S$ determine the numerical properties of the discretization. A general description of the above discretization can be found in *D1* which discusses

the method for obtaining the coefficients a_j. Expression (1) also generalizes to higher derivatives.

The conjunction in two or more dimensions of the above one-dimensional discretization has a small error if the preferential direction points from one grid point to another, following a grid line. In between, the error becomes quite large. The reason is that a derivative which is related to a preferential direction has to be approximated with grid-based derivatives. Grid-based derivatives are using values of the physical unknown(s) which are stored at predefined positions of the grid. A conjunction of one-dimensional discretizations uses only points on the grid lines, and not in between.

This is illustrated in figure 1, where a part of a two-dimensional structured grid is shown. The indices are for the x-direction $0, 1, \ldots, i-1, i, i+1, \ldots, i_{\max}$ and for the y-direction $0, 1, \ldots, j-1, j, j+1, \ldots, j_{\max}$, and the nodal unknowns are indicated by $u_{i,j}$ for node i, j. Indicated are two directions, $\vec{a}_1$ and $\vec{a}_2$. Both point from one grid point to another, but only $\vec{a}_1$ follows a grid line, and the discretization has a small error. For the direction of $\vec{a}_2$, the error is maximal when using a conjunction of one-dimensional discretizations.

The report (*D2*), "Progress in multidimensional upwind differencing", B. van Leer, NASA Contractor Report 189708, ICASE report #92-43, September 1992, NASA Langley Research Center, Hampton, Virginia, describes different known approaches for the simulation of convection phenomena in two dimensions which go beyond one-dimensional methods. In this framework, different state of the art approaches have been proposed to optimize the methods of numerical simulation in two dimensions, with more or less satisfactory performances. Only the methods using an unstructured grid have been exploited in practice using a low order upwind scheme.

In the majority of practical applications, the physical phenomena take place in three dimensional space, which makes it indispensable to realize a numerical simulation of at least three dimensions. In practice, it may be necessary to use more than three dimensions, e.g. to incorporate time as additional dimension, taking into account the fact that an unsteady physical process in N dimensions can be modeled by a $N+1$ dimensional steady process.

However, actually all the industrial methods of numerical simulation in three dimensions (cf. e.g. the documents *D1* and *D2*) are limited to a conjunction of one-dimensional methods.

Besides these one-dimensional methods, certain attempts of truly multi-dimensional numerical simulations in three dimensions have been reported. Nevertheless, taking into account the complexity of the problem presented, and the difficulties encountered in two dimensions, the use of a truly multi-dimensional method in three or more dimensions is

considered to be extremely complex.

The paper (*D3*) "Optimum positive linear schemes for advection in two and three dimensions", P.L. Roe and D. Sidilkover, SIAM J. Num. Anal., December 1992, #6, vol. 29, pages 1542-1568, constitutes one of the rare publications which treat until now a scheme for the simulation of advection phenomena, considering a class of first order upwind schemes. In the case of three dimensions, the authors of the paper, like all authors until now, consider it suitable to choose on a structured grid a minimal upwind stencil formed by a cube of eight points.

Having chosen and predefined this stencil, the different weighting coefficients are computed to obtain the best simulation possible, which is the case of the scheme called "N scheme". Denote the components of the vector $\vec{a}$ of the preferential direction in N dimensions by $a, b, c, \ldots$, that is $\vec{a} = (a, b, c, \ldots)^T$. One-dimensional upwind schemes take the discretization stencil based on $a \geq 0$ or $a \leq 0$. Applying a conjunction of one-dimensional discretization in N dimensions means choosing the stencil depending on the sign of $a, b, c, \ldots$ separately. The N scheme uses instead the combination $a \geq b \geq c \geq \ldots \geq 0$ and points of the stencil which do not lie on a grid line through the point of computation P. Nevertheless, the results obtained with the scheme given in the paper are largely insufficient in practice. Indeed, one cannot obtain a high order scheme with this method on such a small stencil. Furthermore, the simulation in three dimensions generates specific problems which are not encountered in two dimensions.

Moreover, since the date of this paper, no author has effectively imagined that it is possible to obtain higher order accurate discretizations using a larger stencil which includes points beyond the grid lines on a structured grid in three dimensions.

However, as indicated by the publication (*D4*) " Numerical Aerodynamics : Past Successes and Future Challenges from an Industrial Point of View", David Paul Hills, Computational Methods in Applied Sciences, 1996, ECCOMAS, pages 166-173, one-dimensional methods as used in industry have reached their limits, and real gains should be sought in multi-dimensional methods. In spite of continued research since 1992, no such method has been found yet.

Furthermore, the majority of the publications on this subject since 1990 indicate, like the previous, that the true multi-dimensional methods imply upwind discretizations on unstructured grids. Neither multi-dimensional efforts on structured grids nor attempts to find central space discretizations with a reduced error have resulted in numerical simulations which can be applied in an industrial environment.

The invention aims to dispose of these shortcomings by proposing a simulation method which provides a simulated numerical representation in N dimensions, with $N \geq 3$, of spatial p^{th} derivatives D_p, $p \geq 1$, of at least one physical parameter u of a

phenomenon to which is associated a velocity field $\vec{a}$ which determines a preferential direction in every point of at least a part of space, and which permits :

- to reduce significantly the error in the numerical simulation,
- a consistent, stable and convergent simulation,
- a locally monotone simulation,
- the simulation of spatial p^{th} derivatives, with $p \geq 1$, with an error of a desired order, incorporating numerical constraints as desired.

The invention aims also to propose a method which

- is compatible with current technologies, notably computation time and memory requirements,
- is compatible with the simulation of nonlinear phenomena,
- can be used for systems of equations,
- can be used with acceleration techniques like e.g. multi-grid, GMRES and preconditioning,
- and which allows simulations in Finite Element formulations, Finite Volume formulations, Finite Difference formulations, Residual Distribution formulations, Flux Vector Distribution formulation, or others.

The invention also aims at providing a global simulation method of a physical phenomenon and a data processing system, a software, in particular in the form of a digital storage medium, to implement such methods.

SUMMARY OF THE INVENTION

The purpose of this invention is to provide a method of simulating behavior of a flow interacting with an object. At the heart of a numerical simulation is the approximation of space derivatives of a parameter of the flow. An approximation of a space derivative of a parameter involves the use of the value of the parameter at various points at the computational grid. This approximation entails an error, and the larger the error, the larger the cost for a simulation of acceptable accuracy.

The invention improves the accuracy of the approximation of the spatial p^{th} derivative D_p, $p \geq 1$, and therefore reduces the cost of the simulation. It does so by including in the approximation values of the parameter at points which do not lay on grid lines passing through the point of computation P, and by using these values to optimize the approximation. One example is including points on diagonal lines through P, but the invention also describes more general methods. The use of these additional points depends on a preferential direction, such as determined by the advection direction of the flow. The points used in the approximation extend beyond a unit cube on the computational grid. The simulation is in three or more dimensions. The numerical simulation produces output which can be used in the design or optimization of the object which interacts with the flow. In certain cases, it is the output by itself which is important. This is e.g. the case for weather prediction or forest fire simulation.

It is the combination of the extent of the stencil, the use of points which are not on grid lines passing through point P and the optimization of the approximated value D_p^A using the preferential direction which are at the core of the invention. This combination leads to a significant reduction of the error in the approximation of the derivatives, and it allows higher order approximations, which are essential in an industrial context. The improvements of the invention are fully compatible with the current computational practice and aim to dispose of shortcomings of current grid-aligned discretizations by proposing a simulation method which permits :

- to reduce significantly the error in the numerical simulation,
- a consistent, stable and convergent simulation,
- a locally monotone simulation,
- the simulation of spatial p^{th} derivatives, with $p \geq 1$, with an error of a desired order, incorporating numerical constraints as desired.

The invention aims also to propose a method which

- is compatible with current technologies, notably computation time and memory requirements,
- is compatible with the simulation of nonlinear phenomena,
- can be used for systems of equations,
- can be used with acceleration techniques like e.g. multi-grid, GMRES and pre-conditioning,
- and which allows simulations in Finite Element formulations, Finite Volume formulations, Finite Difference formulations, Residual Distribution formulations, Flux Vector Distribution formulation, or others.

The above describes the invention, which aims at improving the numerical tool for the design and the optimization of commercial products, thereby reducing the computational cost of numerical simulations and the time of the design-cycle.

The invention also aims at providing a global simulation method of a physical phenomenon and a data processing system, a software, in particular in the form of a digital storage medium, to implement such methods.

Brief description of the drawings

The following drawings have been included in the description for a better understanding of the invention.

Fig. 1 : Illustration of a part of a two dimensional structured grid with two different advection directions.

Fig. 2 : Perspective view of a material object with a vector of preferential direction.

Fig. 3 : Perspective view of a part of a material surface of an object (wing, hull, ...) in a velocity field $\vec{a}$ (fluid dynamics, electro-magnetics, ...) with a part of a curvilinear grid formed by N ($N = 3$) families of lines, containing points 1-10, and the point of computation P, together with a local basis B which has the basis vector $\vec{e}_1$ aligned with the vector of preferential direction $\vec{a}$.

Fig. 4 : Perspective view of a part of a grid around the origin, showing the coordinate axes emanating from the origin, the 8 corner nodes and the edges of a cube.

Fig. 5 : Perspective view of a part of a grid around the origin, indicating at the 27 nodes of the grid the weights computed by the program described in the appendices for the derivative $(u_x)_{0,0,0}$ used in a second order approximation to a first derivative $D_1 = \partial u/\partial e_1$ on a stencil which does not exceed the cube of figure 4.

Fig. 6 : Perspective view of a part of a grid around the origin, indicating at the 27 nodes of the grid the weights computed by the program described in the appendices for the derivative $(u_y)_{0,0,0}$ used in a second order approximation to a first derivative $D_1 = \partial u/\partial e_1$ on a stencil which does not exceed the cube of figure 4.

Fig. 7 : Perspective view of a part of a grid around the origin, indicating at the 27 nodes of the grid the weights computed by the program described in the appendices for the derivative $(u_z)_{0,0,0}$ used in a second order approximation to a first derivative $D_1 = \partial u/\partial e_1$ on a stencil which does not exceed the cube of figure 4.

Fig. 8 : Perspective view of a part of a grid around the origin, showing the result of the degrees of freedom in the approximation of a first derivative $D_1 = \partial u/\partial e_1$ on a stencil which does not exceed the cube of figure 4, in set 1 of weights at the 27 nodes of the grid as computed by the program described in the appendices.

Fig. 9 : Perspective view of a part of a grid around the origin, showing the result of the degrees of freedom in the approximation of a first derivative $D_1 = \partial u/\partial e_1$ on a stencil which does not exceed the cube of figure 4, in set 2 of weights at the 27 nodes of the grid as computed by the program described in the appendices.

Fig. 10 : Perspective view of a part of a grid around the origin, showing the result of the degrees of freedom in the approximation of a first derivative $D_1 = \partial u/\partial e_1$ on a

stencil which does not exceed the cube of figure 4, in set 3 of weights at the 27 nodes of the grid as computed by the program described in the appendices.

Fig. 11 : Perspective view of a part of a grid around the origin, showing the result of the degrees of freedom in the approximation of a first derivative $D_1 = \partial u/\partial e_1$ on a stencil which does not exceed the cube of figure 4, in set 4 of weights at the 27 nodes of the grid as computed by the program described in the appendices.

Fig. 12 : Perspective view of a part of a grid around the origin, showing the result of the degrees of freedom in the approximation of a first derivative $D_1 = \partial u/\partial e_1$ on a stencil which does not exceed the cube of figure 4, in set 5 of weights at the 27 nodes of the grid as computed by the program described in the appendices.

Fig. 13 : Perspective view of a part of a grid around the origin, showing the result of the degrees of freedom in the approximation of a first derivative $D_1 = \partial u/\partial e_1$ on a stencil which does not exceed the cube of figure 4, in set 6 of weights at the 27 nodes of the grid as computed by the program described in the appendices.

Fig. 14 : Perspective view of a part of a grid around the origin, showing the result of the degrees of freedom in the approximation of a first derivative $D_1 = \partial u/\partial e_1$ on a stencil which does not exceed the cube of figure 4, in set 7 of weights at the 27 nodes of the grid as computed by the program described in the appendices.

Fig. 15 : Perspective view of a part of a grid around the origin, showing the result of the degrees of freedom in the approximation of a first derivative $D_1 = \partial u/\partial e_1$ on a stencil which does not exceed the cube of figure 4, in set 8 of weights at the 27 nodes of the grid as computed by the program described in the appendices.

Fig. 16 : Perspective view of a part of a grid around the origin, showing the result of the degrees of freedom in the approximation of a first derivative $D_1 = \partial u/\partial e_1$ on a stencil which does not exceed the cube of figure 4, in set 9 of weights at the 27 nodes of the grid as computed by the program described in the appendices.

Fig. 17 : Perspective view of a part of a grid around the origin, showing the result of the degrees of freedom in the approximation of a first derivative $D_1 = \partial u/\partial e_1$ on a stencil which does not exceed the cube of figure 4, in set 10 of weights at the 27 nodes of the grid as computed by the program described in the appendices.

Fig. 18 : Perspective view of the three rotations with angles α, β and γ linking the grid with the axes of the local basis B.

Fig. 19 : Perspective view of the subdivision of a unit cube in six tetrahedra.

The drawings illustrate in concrete form specific applications of the invention. The drawings are not meant to exclude from the scope of the invention any application which is the result of any normal and expected modification or extension of the specific application.

Detailed description of the invention

The derivatives according to the invention are called directional derivatives.

The following description uses mathematical symbols, which are explained where introduced, and which, for the purpose of completeness, are summarized in a table at the end of the description.

In the following, a derivative of the form $\partial^p u/\partial e_i^p$ for certain i and p is called a pure derivative, and a derivative of the form $\partial^p u/\partial e_1^{p_1} \partial e_2^{p_2} \ldots$ with $p_1 + p_2 + \ldots = p$ involving partial derivatives along at least two coordinate directions, is called a mixed derivative. The definition is valid for all coordinate systems considered.

A simulation method using directional discretizations of spatial p^{th} derivatives, $p \geq 1$, in N dimensions, $N \geq 3$ on structured grids.

This invention concerns a method of simulating behavior of a flow interacting with an object. The method provides a simulated numerical representation in N dimensions, $N \geq 3$. The representation is composed of a plurality of approximated values at a multitude of points in at least a part of space where the flow interacts with the object. The approximated values are of a physical parameter u of the flow to which is associated a velocity field $\vec{a}$ which determines a preferential direction. The numerical scheme of the simulation approximates at least one spatial p^{th} derivative D_p, $p \geq 1$, of the parameter u at the points of the part of space. The method comprises the following steps :

For the part of space a discrete N-dimensional grid constructed by N families of coordinate lines is used.

In at least one point P of the grid, called the point of computation, an approximated value D_p^A of D_p is computed with an error ϵ_n. This is achieved by using values u_s of the parameter in a collection of grid points. The collection of grid points is called the stencil S. The simulation also uses computational functions C_c, evaluated with the values u_s which depend on the numerical framework in which D_p is expressed.

The computational functions for the approximated value D_p^A are chosen in such a way that the approximated value D_p^A is optimized for the preferential direction.

The stencil S contains at least one point situated outside all the coordinate lines passing through the point of computation P, and the stencil S is chosen in such a way that it contains at least a first point and a second point, the first point being defined by N first coordinate lines of the N families of lines, the second point being defined by N second coordinate lines of the N families of lines, and for at least one family N_f of the coordinate lines, the first coordinate line belonging to the family N_f is different from and not adjacent to the second coordinate line belonging to the same family N_f.

An output is obtained for the numerical representation that simulates, for the part of space, behavior of the flow interacting with the object.

The above describes the invention, which aims at improving the numerical tool for the design and the optimization of commercial products, thereby reducing the computational cost of numerical simulations and the time of the design-cycle. This is achieved according to the invention by reducing the error in the approximation of derivatives, as described above. In certain cases, it is the output by itself which is important. This is e.g. the case for weather prediction or forest fire simulation.

It is the combination of the extent of the stencil, the use of points which are not on grid lines passing through point P and the optimization of the approximated value D_p^A using the preferential direction which are at the core of the invention. This combination leads to a significant reduction of the error in the approximation of the derivatives, and it allows higher order approximations, which are essential in an industrial context, while retaining any of the desired properties mentioned before.

The numerical simulation of physical phenomena is well known, and numerical solutions obtained from computer codes are used in many domains of industry. We will recapitulate the steps which enter into the daily routine of the application of numerical simulations according to the invention. The methodology applies to many domains and applications. The description is an illustration, and is not meant to restrict the invention.

When the decision is taken to use a numerical simulation for the design or optimization of a commercial product, the department responsible for the simulation does the following :

1. A decision is made on which part of physics is important to describe in enough detail the phenomena in a meaningful simulation. In many fluid dynamics cases, the basic fluid dynamics equations like the Navier-Stokes equations are adequate. Perhaps a more simple description can be used. If viscous effects are less important, the Euler equations can be used. At low Mach number, an incompressible approximation can be used. Maybe the potential flow approximation is sufficient. On the other hand, a turbulence model may be needed, together with acoustic models, and a description of the deflection of the object under aerodynamic forces.

The physical description is expressed in a mathematical model comprising of one or more equations which describe the relations between the physically relevant parameters (mass, density, energy, forces, impulse, pressure, volume, temperature, species concentration, charge density, magnetic field, ...). In general, the mathematical model uses spatial derivatives of one or more of the parameters, as well as time derivatives. Together with the number of space dimensions, the physical model determines the number

of variables which are needed in the solution. In the case of the Navier-Stokes equations in three dimensions, five variables which depend on space and time describe the physical solution. These five variables can be e.g. the density, the momentum in the three directions and the energy. This set of variables is commonly known as conservative variables. Other combinations are possible, such as the density, the three velocity components and the pressure, which is known as the set of primitive variables. Yet another set of variables is known as entropy variables, since one of the variables is chosen to be proportional to the entropy. There are still many other sets of independent variables. A transformation is always possible between different sets of independent variables.

2. The next step is to choose a computer program which can compute a solution to the problem which is compatible with the physical description of the problem. The simulation code will be run on a computer. While the physics and the mathematical model describe a continuum, the computer can only work with discrete values in a limited number of points. The solution of the mathematical model with a computer program therefore involves the procedure of discretization, which consists of two steps.

First, the continuous variable is replaced by a discrete variable in space. A grid defines points in space where the values of the parameters are stored.

The second step is to rewrite the relations between the continuous variables of the mathematical model in relations between discrete variables. In particular, the derivatives which appear in the mathematical model are replaced by their discrete equivalent, which are differences which are locally calculated using the values of the discrete variables of the surrounding grid points. This step involves an approximation and is not exact.

In general, a big enough company has the means and experience to develop and maintain an in-house computer code. Such a code is tuned to the specific needs of the company. It may compute solutions to only a very restricted number of physical problems, but it is likely to be up to date. The developments in the domain are followed, and e.g. new space discretizations, faster solution techniques such as accelerators and so on are coded. After testing, the code is ready for production runs. Smaller enterprises often can afford only a small computational group, which uses programs from a specialized provider. This is normally a more general code, since it will have to accommodate a larger spectrum of clients with different applications.

3. The computation of the numerical solution is preceded by a pre-processing step. A geometrical description of the object is used to generate a computational grid. This grid defines the points in space where the solution is calculated. Together with this grid an initial solution is generated. In the case of the Navier-Stokes equations, the five independent variables are given a certain value for each of the grid points, corresponding

to e.g. uniform flow at the design Mach number. The grid and the initial solution are written to one or more files.

4. The next step is to run the simulation code. The program performs the following actions :

a) The program reads the grid, and stores for each point the coordinates in the memory of the computer.

b) It reads the initial solution and stores for each grid point the value of the physical parameters in memory.

c) Additional files are read which specify computational parameters such as the number of iterations or a convergence criterion, which physical model to use, which of the coded space discretizations to use, and additional information if necessary. Boundary conditions are specified which are used at the extremities of the part of space under consideration. They represent inflow and outflow from the domain, the way the object influences the flow, and any other interaction of the flow with its surroundings.

d) Using the grid, the initial solution and the boundary conditions, it solves iteratively and approximately the discrete mathematical equations which describe the physical problem. At the core of the numerical simulation is the approximation of space derivatives. The invention provides improved discretization with a reduced error.

e) The code stops after a specified number of iterations, or when a solution is obtained which is accurate enough.

f) The code writes the solution to a file. This output solution consists of the value of physical variables in each of the grid points, where the number of physical variables corresponds to the problem. Another output file may be written which contains the grid.

5. A post-processing program is used to read the grid file and the output file of step 4f. Normally this very general program permits to extract any desired physical parameter which can be computed from the solution. If the solution is written in the form of conservative variables, this program can compute the pressure, which is needed for e.g. the lift and drag. It can also compute variables like the Mach number, the temperature and so on.

6. The results are used to modify or improve the geometry.

Steps 3-6 are repeated until the design is optimal. In the case of insufficiency of the physical model, a more elaborate one is chosen, and coded if necessary. If there is insufficient resolution of the simulated physical phenomena on parts of the grid, a finer grid is generated.

The above describes the typical computation of a numerical solution and its use in an industrial environment. A similar procedure is followed when the purpose is to design or to optimize a part of an aircraft, a car, a ship, a vehicle in general, a turbo machine, a ventilation system, a wind mill, a pump, a combustion engine, a channel, the mixture of steam and oil or water and oil in porous media for oil recovery, and so on, or to describe the evolution of flow such as in weather prediction, climate modeling or in a forest fire. In step 1, the physical model depends on the problem under consideration. In a magneto-hydrodynamics application, the description of the electro-magnetic field enters. This entails the use of additional variables in the solution. In an application with different chemical species, the concentration of each of the species enters, together with a description of the transformation between the different chemical compounds. In multi-phase flow, such as e.g. oil and steam, or water and air, all fluids require equations and variables for a proper description.

The pre-processing and post-processing, steps 3 and 5 respectively, are normally separate programs. Occasionally, they can be incorporated in the numerical solver.

In step 3 the grid is generated once for a given geometry. In current practice a grid can contain many millions of points. Very large computational grids are normally composed of different blocks. Such a multi-block grid can then be distributed to multiple processors or even to different computers for a parallel computation. Some numerical simulation programs are able to refine or derefine automatically the grid locally using an error estimation based on the computed solution. Some programs use such a capability to compute a time-accurate simulation where a concentration of grid points follows a physical phenomena like e.g. a vortex or a shock. In the case of a time-accurate simulation, step 4f is repeated for each intermediate solution of interest. This is e.g. the case for the instationary behavior of a turbo machine where the radiated sound is computed. Another example is combustion.

The practical use of numerical simulations is illustrated in step 5, where the output of the program is transformed into a useful result. The output of the numerical simulation can serve many purposes, depending on the application. Quite often, this involves the pressure, since this relates to forces on the object under design. For a wing, this can be the Mach number for restricting the shock strength. For turbo machines, this can be the pressure on the blades, or the thermal load. For a heart valve, this can be the velocity field which needs to be smooth to avoid cluttering. For a forest fire simulation,

this is again the velocity field to predict the likely progress of the fire.

The simulated numerical representation may be the values of the physical parameter, or the variations of the physical parameter in time or in space, or any combination of the physical parameter. The simulated numerical representation can be used in the design or optimization of an object by a transformation of the representation, using post-processing tools, e.g. into a film or video such as showing the evolution of a flow, especially unsteady effects such as buffeting, or an image such as a flow field, or a curve such as lift versus drag of a wing, or a coefficient such as the overall drag, or in general any transformation as required by the application under consideration.

Normally, the person who runs the program makes use of the output as illustrated in step 6. The output can also be fed back without human interference into the program. This is e.g. the case for inverse methods, where the desired pressure distribution is prescribed in the form of a goal. The geometry of the object and the related grid possess degrees of freedom which allow optimization to reach the goal. Remark that the purpose in the example above is the optimal geometry of the object. The numerical solution is a convenient tool to achieve this goal.

A three-dimensional example is given in figure 2, where a solid surface is exposed to a flow. The surface can be e.g. a wing, a blade of a propeller, part of a duct, the hull of a ship or submarine, a part of an aircraft or a car, or in general the surface of any object encountered in an industrial context which needs to be designed or optimized. The velocity is indicated by the vector $\vec{a}$.

The design or optimization with a numerical tool implies an approximation of derivatives of the physical parameter u. Such a derivative is e.g. a stream-wise first derivative as encountered in flow problems, $D_1 = \vec{a}.\vec{\nabla}u$, where $\vec{\nabla}$ is the N-dimensional gradient, $\vec{\nabla}u = (u_x, u_y, u_z, \ldots)^T$, which is composed of grid-based derivatives. The invention concerns the approximation D_p^A using the combination of grid-based derivatives u_x, u_y, u_z,... and other combinations for the approximation of higher derivatives with a reduced error.

For the approximation of derivatives, the tool has at its disposition the approximated values u_s stored at certain positions in space, according to a grid as indicated in figure 3, which is a blow-up of a part of figure 2.

The points 1, P and 2 lie on a grid line of one family, while the points 3 and 7 and the points 4 and 9 lie on different lines of the same family. The points 3, P and 4 lie on a grid line of another family, while the points 1 and 9 and the points 2 and 7 lie on different lines of the same family of lines as points 3, P and 4. Finally, the points 5, P and 6 lie on a grid line of yet another family, which family is shared by the points 7 and 8 and the points 9 and 10, although the latter are on different grid lines.

Especially, in figure 3, point P indicates a position where the numerical tool needs a derivative approximation. This derivative approximation is a function of the values of u_s at the nodes of the stencil S which are used to approximate the derivative.

The precise way in which the unknowns u_s are used in the computation of the derivate depends on the numerical framework which is used in the numerical tool. In a given numerical framework, this depends on computational functions specific to that framework. In the case of the Finite Difference framework, the computational functions are coefficients, and the values of the physical parameter u_s at the points of the stencil S is multiplied with its coefficient, and the linear combination of values constitute the approximation of the derivative. In the case of the Finite Volume framework, the fluxes through the surfaces of a volume around P depend on the values of the physical parameter u_s at the points of the stencil S. These and other formulations are described in more detail below, but in general, the approximation of the derivative depends on the values u_s at the points of the stencil S through the use of computational functions which depend on the numerical framework used.

Since the approximation of the derivative depends on the values u_s at the points of the stencil S, an expansion of each of the values u_s into a truncated Taylor series with respect to the point P is instrumental in obtaining an estimation of the error of the approximation of the derivative. The use of Taylor series is especially useful when optimizing an approximation of a derivative with a prescribed order of accuracy when the maximum extent of the stencil is given. This is often the case for the numerical tool used in industry. Given these conditions, it is possible to make an inventory of all the consistent approximations of the grid-based derivatives involved in D_p^A. This general approximation of each of the grid-based derivatives takes the form of a sum of stencils. One of the stencils in the sum represents a particular consistent approximation to the grid-based derivative, while each of the remaining stencils in the sum appears with a coefficient which represents a degree of freedom. The stencils associated with a degree of freedom are approximations which are only apparent in D_p^A in the leading error term or beyond. The coefficients can be constants, or a function of physical parameters, such as the preferential direction, e.g. through the angles of the transformation which links the basis B with the grid. In the latter case, the approximation of D_p^A becomes also dependent on the angles.

The discretizations for first and higher derivatives are constructed in a systematic way as described above, which is very well suited for an implementation in a computer program. Such a program is joined in appendices 1-5 to the instant specification.

An alternative is the formulation of Hildebrand, see *D1*, which however needs to be generalized for use in N dimensions. Such a generalization has been obtained, and leads

to the same results.

In the following, the various terms involving sums of stencils will be indicated by T, possibly with a subscript to indicate a particular grid-based derivative. An example is given in appendix 6, where a general approximation is described in three dimensions to a stream-wise first derivative which is second order accurate on a grid around P with an extent of three points in each direction, i.e. containing $3^3 = 27$ nodes. The grid is similar to the one shown in figure 3. The grid-based derivatives which are needed in the approximation of D_1^A, u_x, u_y and u_z are each expressed as just described. There are 17 degrees of freedom for each grid-based derivative, as indicated by the coefficients $k_{x1} \ldots k_{x17}$ for u_x, and similar for the other derivatives. Other examples are given further on.

Already on a small grid, the number of degrees of freedom is large. On more extended grids, this number increases rapidly, and the newly found degrees of freedom open a vast region of optimization and allow for important error reductions. The complete set of stencils, representing all the degrees of freedom is an integral part of the invention.

The leading error term can be a guideline in the choice of the stencil S for optimizing the derivative. As an example, the parts of the error which are related to directions normal to the preferential direction can be eliminated, partially or totally, for certain or all directions. The derivative can also be optimized using a representation of the numerical solution in Fourier components. The use of Fourier components permits to analyze a discretization in terms of stability and convergence. Therefore, the optimization can be according to criteria concerning stability, convergence, damping of high-frequency components which are important for multi-grid acceleration, dispersion of the discretization, important for the behavior of wave propagation as encountered in acoustic applications, and so on and so forth. The degrees of freedom described just above are particularly appropriate for such an optimization. The optimization of the derivatives is for certain preferential directions. Remark that the optimization of the error or of Fourier components are just two examples. Other optimization criteria are effectively any numerical property of the discretization, as explained in *D1*, including symmetry of the resulting stencil, a minimum number of contributing nodes to the stencil for computational cost, ease of implementation, and combinations thereof. It is to be noted that adding a discretization of sufficiently high order to the approximation of the derivative does not harm the optimization.

As an example, for the preferential direction $\vec{a}$ indicated in figure 3, the error of the numerical simulation due to the approximation of the derivative can be significantly reduced by the use of an approximation which uses the stencil S, given by the list of the nodes of which it consists, $S = \{P, 1, 2, 7, 8, 9, 10\}$. Remark that the nodes 7-10 do not

lie on grid lines passing through the point of computation P. Furthermore, the stencil S is extended in the sense that it does not fit in a unit hypercube of the grid. An example of a unit hypercube of the grid is the cube which has as constituting edges the three edges P-2, P-4 and P-6.

By using the approximations of derivatives according to the invention, the error is reduced, the numerical tool is improved and the computational effort is reduced. In a typical industrial application, the numerical tool may use an implicit fourth order method for the approximation of derivatives. Using the invention, the average error can then be reduced by a factor of 4, and the same quantitative simulation can be obtained for one fifth of the computational cost.

As an example, some of the discretizations are described below, optimizing the error or optimizing Fourier components, in the Finite Difference formulation and in the Finite Element formulation, on a rectilinear grid, but the scope of the invention is not limited to those. It should be equally understood that the invention can be used on grids which are normally encountered in industrial applications of numerical simulations. Such grids can be perfectly regular, or contain distortions, they can be static, moving, deforming or rotating. The invention applies to staggered and to non-staggered grids, single block or multi block, and any combination of the aforementioned characteristics.

A simulation method using continuous directional discretizations of spatial p^{th} derivatives, $p \geq 1$, in N dimensions, $N \geq 3$ on structured grids.

The discretizations of the invention depend on the preferential direction. This means that the stencil for an approximation involves different sets of points. For one direction, one set of points is used in the approximation, while for another direction a different set is used. Examples are given further on.

It is important that the approximation D_p^A depends continuously on the values u_s. In other words, the stencil should not change discontinuously for a small variation of the preferential direction. Continuity of the approximation is important for the convergence of the numerical method.

A simulation method using directional discretizations of spatial p^{th} derivatives, $p \geq 1$, in N dimensions, $N \geq 3$ on structured grids, optimizing the discretization using a local basis.

The above derivatives can be optimized using a local basis. This means that a local basis $B(\vec{e}_1, \vec{e}_2, \vec{e}_3, \ldots)$ of curvilinear coordinates is used which has the unit vector $\vec{e}_1$ along the preferential direction. In the approximated value D_p^A, the computational functions are chosen in such a way that the contribution to the error ϵ_n of at least one pure or one mixed derivative as expressed in the local basis B is minimized, while using as a formulation of the values u_s of the parameter at each of the points of the stencil S, a

truncated Taylor series expansion with respect to the point of computation P with an error, called the truncation error ϵ_s.

When the numerical tool needs to approximate a derivative which is related to the preferential direction $\vec{a}$, a local basis B is used which has a basis vector e_1 aligned with $\vec{a}$ as indicated in figure 3. When the error in the approximation of the derivative is expressed in this local basis B, some pure or mixed derivatives can be minimized by an appropriate choice of the stencil, and by the choice of the computational functions which are used in the numerical framework.

A simulation method using directional discretizations with Finite Differences and a local basis.

The above directional discretizations can be used in the framework of Finite Differences, where the approximation D_p^A is a linear combination D_p^{LC} of the values of the parameter(s) u_s in a collection of grid points, called stencil S, with computational functions which are individual coefficients $C_s = C_{l,m,n,...}$. This means that

$$D_p^{LC} = \sum_{l,m,n,...\in S} C_{l,m,n,...} u_{l,m,n,...} \,. \tag{2}$$

In the linear combination D_p^{LC} the coefficients C_s are chosen in such a way that the contribution to the error ϵ_n of at least one pure or one mixed derivative as expressed in the local basis B is minimized.

A simulation method using directional discretizations with Finite Volumes and a local basis.

The above directional discretizations can be used in the framework of Finite Volumes. This formulation is very closely related to Finite Differences, and is based on an integral formulation using computational functions f which are fluxes through a control volume. These fluxes are in turn a function of a number of the nodal unknowns, defining again a stencil. This in turn leads to an approximation which has similarities with the expression of equation 2. In the approximation D_p^A the fluxes are chosen in such a way that the contribution to the error ϵ_n of at least one pure or one mixed derivative as expressed in the local basis B is minimized.

It should be understood that the invention applies to the cell-centered Finite Volume method as well as to the vertex-centered Finite Volume method.

A simulation method using directional discretizations with Finite Elements and a local basis.

The above directional discretizations can be used in the framework of the Finite Element method, where the space is subdivided into elements, containing nodes at which the approximation of the physical parameter(s) u is stored. Basis computational functions ϕ, called interpolation functions, are defined on the elements, which are used to approxi-

mate the physical value u on the element. The integral of the derivative is computed on the element, using a test computational function ψ, called weighting function. The approximated value $D_p^A = D_p^{\phi,\psi}$ is expressed in the nodal unknowns u_s at a collection of points, called stencil S. In the approximated value $D_p^{\phi,\psi}$, the combination of ϕ and ψ is then chosen in such a way that the contribution to the error ϵ_n of at least one pure or one mixed derivative as expressed in the local basis B is minimized.

A simulation method using directional discretizations with the Residual Distribution Method and a local basis.

The above directional discretizations can be used in the framework of the Residual Distribution Method. This method shares with Finite Elements the representation of the unknowns u on the element, and with the Finite Volume method a volume over which the integral of the derivative is computed. The integral I_{el} of the derivative is split into parts $\alpha_i I_{el}$ which are distributed over the nodes i. The approximated value $D_p^A = D_p^{\alpha_i}$ is expressed in the nodal unknowns u_s at a collection of points, called stencil S. In the approximated value $D_p^{\alpha_i}$, the distribution computational coefficients α_i are then chosen in such a way that the contribution to the error ϵ_n of at least one pure or one mixed derivative as expressed in the local basis B is minimized.

A simulation method using directional discretizations with the Flux Vector Distribution Method and a local basis.

Another Distribution method is the Flux Vector Distribution method. Here, the representation of the unknowns is the same as for the Residual Distribution method, but now the flux f through surfaces of volumes are split into parts $\alpha_i f$ which are distributed over the nodes i. The approximated value $D_p^A = D_p^{\alpha_i}$ is expressed in the nodal unknowns u_s at a collection of points, called stencil S. In the approximated value $D_p^{\alpha_i}$, the distribution computational coefficients α_i are then chosen in such a way that the contribution to the error ϵ_n of at least one pure or one mixed derivative as expressed in the local basis B is minimized.

A simulation method using directional discretizations of spatial p^{th} derivatives, $p \geq 1$, in N dimensions, $N \geq 3$ on structured grids, optimizing the discretization using Fourier components.

The above derivatives can be optimized using Fourier components. This means that a representation of the numerical solutions in Fourier components is used. In the approximated value D_p^A, the computational functions are chosen in such a way that the Fourier components are optimized for certain directions which are related to the velocity $\vec{a}$, while using the values u_s of the parameter at each of the points of the stencil S in the Fourier representation.

The use of Fourier components is clearly described in *D1*. Any finite mesh function,

such as the solution or the error in the solution can be decomposed into a Fourier series. The solution $u(\vec{x}, t)$ is then a superposition of harmonics of the form

$$u(\vec{x}, t) \sim A e^{-I\omega t} e^{I\vec{\kappa}.\vec{x}}. \tag{3}$$

The following notation is used : $\vec{x}$ for the position, t for time, A an amplitude, I the imaginary unit, such that $I^2 = -1$, ω the angular frequency, and $\vec{\kappa}$ the wave number vector. These terms are explained in *D1*. These harmonics can be substituted in a Finite Difference representation of the discretization. This permits to analyze numerical properties of the scheme, such as stability and convergence, wave propagation and so on. The harmonics of equation 3 depend on the position vector $\vec{x}$, which is different for each of the points of the stencil S.

The numerical properties as established by using Fourier components depend on the computational functions, and therefore these functions may be chosen in such a way that certain numerical properties are optimized. This can be done for certain preferential directions as indicated in figure 3. Like mentioned before, for the preferential direction $\vec{a}$, certain numerical properties like stability will be optimized by the use of an approximation which uses the stencil S, given by the list of the nodes of which it consists, $S = \{P, 1, 2, 7, 8, 9, 10\}$.

A simulation method using directional discretizations with Finite Differences and Fourier components.

The above directional discretizations can be used in the framework of Finite Differences, where the approximation D_p^A is a linear combination D_p^{LC} of the value of the parameter u_s in a collection of grid points, called stencil S, with computational functions which are individual coefficients $C_s = C_{l,m,n,...}$, see equation 2. In the linear combination D_p^{LC} the coefficients C_s are chosen in such a way that the Fourier components are optimized for certain directions which are related to the velocity $\vec{a}$, while using the value u_s of the parameter at each of the points of the stencil S in the Fourier representation.

A simulation method using directional discretizations with Finite Volumes and Fourier components.

The above directional discretizations can be used in the framework of Finite Volumes. This formulation is very closely related to Finite Differences, and is based on an integral formulation using computational functions f which are fluxes through a control volume. These fluxes are in turn a function of a number of the nodal unknowns, defining again a stencil. This in turn leads to an approximation which has similarities with the expression of equation 2. In the approximation D_p^A the fluxes are chosen in such a way the Fourier components are optimized for certain directions which are related to the velocity $\vec{a}$, while using the value u_s of the parameter at each of the points of the stencil S in the Fourier representation.

It should be understood that the invention applies to the cell-centered Finite Volume method as well as to the vertex-centered Finite Volume method.

A simulation method using directional discretizations with Finite Elements and Fourier components.

The above directional discretizations can be used in the framework of the Finite Element method, where the space is subdivided into elements, containing nodes at which the approximation of the physical parameter(s) u is stored. Basis computational functions ϕ, called interpolation functions, are defined on the elements, which are used to approximate the physical value u on the element. The integral of the derivative is computed on the element, using a test computational function ψ, called weighting function. The approximated value $D_p^A = D_p^{\phi,\psi}$ is expressed in the nodal unknowns u_s at a collection of points, called stencil S. In the approximated value $D_p^{\phi,\psi}$, the combination of ϕ and ψ is then chosen in such a way that the Fourier components are optimized for certain directions which are related to the velocity $\vec{a}$, while using the value u_s of the parameter at each of the points of the stencil S in the Fourier representation.

A simulation method using directional discretizations with the Residual Distribution Method and Fourier components.

The above directional discretizations can be used in the framework of the Residual Distribution Method. This method shares with Finite Elements the representation of the unknowns u on the element, and with the Finite Volume method a volume over which the integral of the derivative is computed. The integral I_{el} of the derivative is split into parts $\alpha_i I_{el}$ which are distributed over the nodes i. The approximated value $D_p^A = D_p^{\alpha_i}$ is expressed in the nodal unknowns u_s at a collection of points, called stencil S. In the approximated value $D_p^{\alpha_i}$, the distribution computational coefficients α_i are then chosen in such a way that the Fourier components are optimized for certain directions which are related to the velocity $\vec{a}$, while using the value u_s of the parameter at each of the points of the stencil S in the Fourier representation.

A simulation method using directional discretizations with the Flux Vector Distribution Method and Fourier components.

Another Distribution method is the Flux Vector Distribution method. Here, the representation of the unknowns is the same as for the Residual Distribution method, but now the flux f through surfaces of volumes are split into parts $\alpha_i f$ which are distributed over the nodes i. The approximated value $D_p^A = D_p^{\alpha_i}$ is expressed in the nodal unknowns u_s at a collection of points, called stencil S. In the approximated value $D_p^{\alpha_i}$, the distribution computational coefficients α_i are then chosen in such a way that the Fourier components are optimized for certain directions which are related to the velocity $\vec{a}$, while using the value u_s of the parameter at each of the points of the stencil S in the

Fourier representation.

A simulation method using directional discretizations on orthogonal curvilinear coordinate systems.

The most commonly used curvilinear coordinate systems are orthogonal, such as : rectangular coordinates (also called rectilinear coordinates), spherical coordinates, cylindrical coordinates, parabolic cylindrical coordinates, paraboloidal coordinates, elliptic cylindrical coordinates, prolate spheroidal coordinates, oblate spheroidal coordinates, bipolar coordinates, toroidal coordinates, conical coordinates, confocal ellipsoidal coordinates or confocal paraboloidal coordinates, see (*D5*) "Mathematical Handbook", Murray R. Spiegel, McGraw-Hill, New York, 1968. The above directional discretizations can be used therefore in those coordinate systems, where the expression of nodal values u in a truncated Taylor series expansion, and the transformation to the local coordinate system $B(\vec{e_1}, \vec{e}_2, \vec{e}_3, \ldots)$, as well as the Fourier representation and other numerical properties depend on the specifics of the structured grid which is used.

A simulation method using directional discretizations on a wide variety of grids.

The invention can be applied to the most basic grids with uniform mesh spacing which are fixed in space, but equally to grids which have non-uniform mesh spacing, which are moving, which are deforming, which are rotating, which are staggered, and any combination thereof.

A simulation method using directional discretizations in rectilinear coordinate systems.

The above directional discretizations can be used in rectilinear coordinate systems. In the case of regular rectangular coordinates, the expression of nodal values u in a truncated Taylor series means that for a certain point u_s with indices $(l, m, n, \ldots)$,

$$u_{l,m,n,\ldots} = u_{i,j,k,\ldots} + \sum_{r=1}^{r_{\max}} \frac{1}{r!} \left((l-i)\Delta x \frac{\partial}{\partial x} + (m-j)\Delta y \frac{\partial}{\partial y} + (n-k)\Delta z \frac{\partial}{\partial z} + \ldots \right)^r u \, , \tag{4}$$

where the error in $u_{l,m,n,\ldots}$ depends on the last term in the series, which is determined by the upper limit in the summation, $r_{\max}$. A term of the form $(q1 + q2 + q3 + \ldots)^r$ can be written as

$$(q_1 + q_2 + q_3 + \ldots)^r = \sum_{n_1, n_2, n_3, \ldots} \frac{r!}{n_1!\, n_2!\, n_3! \ldots} q_1^{n_1} q_2^{n_2} q_3^{n_3} \ldots \, , \tag{5}$$

where the sum is over all nonnegative integers $n_1, n_2, n_3, \ldots$ for which $n_1 + n_2 + n_3 + \ldots = r$, see *D5*.

The partial derivatives appearing in the truncated Taylor series expansions can be

rewritten in derivatives of the local basis B using the coordinate transformation

$$\begin{array}{rcl} x &=& t_{1,1}e_1 + t_{1,2}e_2 + t_{1,3}e_3 + \dots \\ y &=& t_{2,1}e_1 + t_{2,2}e_2 + t_{2,3}e_3 + \dots \\ z &=& t_{3,1}e_1 + t_{3,2}e_2 + t_{3,3}e_3 + \dots \\ \vdots & & \vdots \qquad \vdots \qquad \vdots \qquad \ddots \end{array} , \qquad (6)$$

where the coefficients $t_{\alpha,\beta}$ depend on the angles between the two coordinate systems.

A simulation method using directional discretizations for pure derivatives.

A special case of the above space discretizations is obtained for pure derivatives, $\partial^p u/\partial e_i^p$ for certain i and p. These are first and higher pure derivatives. For $i = 1$, they are along the preferential direction. The computational functions are chosen in such a way that the approximated value D_p^A is optimized for the preferential direction.

A simulation method using directional discretizations for the first derivative.

A special case of the above space discretizations is obtained for $p = 1$, which refers to the first derivatives, which are also called gradients. These derivatives are the building block for any numerical simulation of advection phenomena.

A simulation method using directional discretizations for the second derivative.

A special case of the above space discretizations is obtained for $p = 2$, which refers to the second derivatives. Second derivatives appear in the discretization of viscous terms, or in a purely numerical fashion as artificial viscosities. These derivatives are the building block for any numerical simulation including viscosity.

A simulation method using directional discretizations for mixed derivatives.

A special case of the above space discretizations is obtained for mixed derivatives, $\partial^p u/\partial e_1^{p_1} \partial e_2^{p_2} \dots$ with $p_1 + p_2 + \dots = p$. The computational functions are chosen in such a way that said approximated value D_p^A is optimized for the preferential direction.

A simulation method using directional discretizations for the first derivative $D_1 = \partial u/\partial e_1$ using points on diagonals.

A special case of the above directional discretizations is using the approximation given by

$$\begin{pmatrix} (u_x)_{i,j,k,\dots} \\ (u_y)_{i,j,k,\dots} \\ (u_z)_{i,j,k,\dots} \\ \vdots \end{pmatrix} , \qquad (7)$$

wherein

$$(u_x)_{i,j,k,\ldots} = \frac{1}{\Delta x}\Big\{ \; a_{-m}u_{i-m,j,k,\ldots} + a_{-m+1}u_{i-m+1,j,k,\ldots} + \cdots$$
$$+ a_{n-1}u_{i+n-1,j,k,\ldots} + a_n u_{i+n,j,k,\ldots} + Tx \Big\} ,$$
$$(u_y)_{i,j,k,\ldots} = \frac{1}{\Delta y}\Big\{ \; a_{-m}(u_{i-m,j-m,k,\ldots} - u_{i-m,j,k,\ldots})$$
$$+ a_{-m+1}(u_{i-m+1,j-m+1,k,\ldots} - u_{i-m+1,j,k,\ldots}) + \cdots$$
$$+ a_{n-1}(u_{i+n-1,j+n-1,k,\ldots} - u_{i+n-1,j,k,\ldots})$$
$$+ a_n(u_{i+n,j+n,k,\ldots} - u_{i+n,j,k,\ldots}) + Ty \Big\} ,$$
$$(u_z)_{i,j,k,\ldots} = \frac{1}{\Delta z}\Big\{ \; a_{-m}(u_{i-m,j-m,k-m,\ldots} - u_{i-m,j-m,k,\ldots})$$
$$+ a_{-m+1}(u_{i-m+1,j-m+1,k-m+1,\ldots} - u_{i-m+1,j-m+1,k,\ldots}) + \cdots$$
$$+ a_{n-1}(u_{i+n-1,j+n-1,k+n-1,\ldots} - u_{i+n-1,j+n-1,k,\ldots})$$
$$+ a_n(u_{i+n,j+n,k+n,\ldots} - u_{i+n,j+n,k,\ldots}) + Tz \Big\} ,$$
$$\vdots \tag{8}$$

wherein $a_{-m} \neq 0$, $a_n \neq 0$, m and n are given integers, $m+n > 0$, and $m+n > 1$ if $m*n = 0$, the terms Tx, Ty, Tz, ... represent the degrees of freedom which are used in the optimization of said approximated value D_1^A, and where indices $(i, j, k, \ldots)$ define the point of computation P on the N-dimensional grid, and $\Delta x, \Delta y, \Delta z, \ldots$ denote the mesh spacings of the N-dimensional grid in each coordinate direction..

The combination of the first derivatives of equations 7 is also called gradient in N dimensions.

Without the terms Tx, Ty, Tz, ..., this approximation amounts to using a one-dimensional discretization for the x-derivative. This is normally a consistent approximation to a first derivative, with numerical properties such as the desired order. Then, the same one-dimensional discretization is applied to the diagonal nodes in the $x-y$-plane to obtain the sum of the x and y derivatives, from which the y derivative follows. This construction is applied consequently for each additional space dimension. This approximation has therefore a reduced error when the preferential direction $\vec{a}$ is directed along the x-axis, and along certain diagonals of the grid. Remark that when m and n are of opposite sign, the stencil does not contain the point of computation P. For m and n positive or zero, P is part of the stencil, but appears only in the x-derivative.

With the terms Tx, Ty, Tz, ... included, all the consistent approximations on the grid with an extent of $(-m..n)$ in each grid direction can be found, as explained before. The terms T, which represent the degrees of freedom in the form of stencils with a multiplying coefficient, can be used for any optimization.

This discretization has a reduced error for preferential directions with $a \geq b \geq c \geq \ldots \geq 0$, and permutations thereof for other directions, but the discretization can be applied to other preferential directions.

As mentioned before, the stencils corresponding to the degrees of freedom, indicated by T, can be chosen in such a way that the approximation depends continuously on the preferential direction. Without the terms T, the discretization is already continuous.

As an example, consider the case $m = n = 1$, $a_1 = \frac{1}{2}$, $a_0 = 0$ and $a_{-1} = -\frac{1}{2}$ in three dimensions, and take for simplicity the point P at the origin, which means $i = j = k = 0$. These values substituted in equations 8 result in the second order directional central discretization, which is given by

$$\begin{aligned}
(u_x)_{0,0,0} &= \frac{1}{\Delta x}\left\{\frac{1}{2}(u_{1,0,0} - u_{-1,0,0}) + Tx\right\}, \\
(u_y)_{0,0,0} &= \frac{1}{\Delta y}\left\{\frac{1}{2}(u_{1,1,0} - u_{1,0,0} + u_{-1,0,0} - u_{-1,-1,0}) + Ty\right\}, \\
(u_z)_{0,0,0} &= \frac{1}{\Delta z}\left\{\frac{1}{2}(u_{1,1,1} - u_{1,1,0} + u_{-1,-1,0} - u_{-1,-1,-1}) + Tz\right\}.
\end{aligned} \tag{9}$$

A permutation of this stencil is given in figure 3. If the x-axis is taken parallel to the direction given by nodes 1 and 2, the y-axis is taken parallel to the direction given by nodes 3 and 4 and the z-axis is taken parallel to the direction given by nodes 5 and 6, the x-derivative involves the nodes 1 and 2, the y-derivative involves the nodes 7, 2, 1 and 9, and the z-derivative involves the nodes 8, 7, 9 and 10. The stencil of this discretization fits in the cube shown in figure 4. Figure 4 shows the coordinate system as well as eight corner points on a grid ranging from -1 to 1 in all directions, defining a cube. The origin is at the intersection of the three axes.

In the appendices 1-5, the listing of a program is given which computes in a systematic way approximations in three dimensions. The user defines in an input file the derivative to be approximated, the order of accuracy and the maximum extent of the stencils. The program is used with this specification to generate all possible stencils which satisfy these constraints. The resulting stencils are in the form like presented in equations 8, a stencil which satisfies the constraints and terms T which represent stencils which can be added without violating the constraints. The terms T can be added since they appear in the discretization in the leading error term. The terms T form a basis of stencils with which all stencils can be formed which satisfy the constraints. Also an optimization parameter needs to be given to the program which computes discretizations. Without optimization, all the stencils giving a consistent approximation are generated, which can then be used for consequent optimization. One particular optimization which is coded reduces the approximation to a pure one-dimensional discretization when the preferential direction is along the x-axis or along diagonals. This optimization therefore

reduces the number of degrees of freedom.

For the latter optimization, the computer-generated images using the results of the program as presented in appendix 8 are shown in figures 5-17 for a second order approximation to a first derivative $D_1 = \partial u/\partial e_1 = \vec{e}_1.\vec{\nabla} u$ on a stencil which does not exceed the cube of figure 4. The optimization has left ten degrees of freedom for each grid-based derivative, which is a reduction from the 17 degrees of freedom without optimization, see appendix 6.

The meaning of these figures is as follows. In figures 5, 6 and 7, the stencils Tfx, Tfy and Tfz of appendix 8 are shown which are used in the approximation of $(u_x)_{0,0,0}$, $(u_y)_{0,0,0}$ and $(u_z)_{0,0,0}$ respectively, which corresponds to

$$\begin{aligned}
(u_x)_{0,0,0} &= \frac{1}{2\Delta x}(2u_{0,0,0} - 2u_{-1,0,0} + u_{1,-1,0} - 2u_{0,-1,0} + u_{-1,-1,0} \\
&\qquad + u_{1,0,-1} - 2u_{0,0,-1} + u_{-1,0,-1} \\
&\qquad - u_{1,-1,-1} + 2u_{0,-1,-1} - u_{-1,-1,-1}) , \\
(u_y)_{0,0,0} &= \frac{1}{2\Delta y}(2u_{0,0,0} - 2u_{-1,0,0} + u_{-1,1,0} - 2u_{0,-1,0} + u_{-1,-1,0} \\
&\qquad + 2u_{0,1,-1} - 2u_{-1,1,-1} + u_{1,0,-1} - 6u_{0,0,-1} + 5u_{-1,0,-1} \\
&\qquad - u_{1,-1,-1} + 4u_{0,-1,-1} - 3u_{-1,-1,-1}) , \\
(u_z)_{0,0,0} &= \frac{1}{2\Delta z}(2u_{-1,0,1} + 2u_{0,-1,1} - 3u_{-1,-1,1} \\
&\qquad + u_{-1,1,0} + 4u_{0,0,0} - 8u_{-1,0,0} \\
&\qquad + u_{1,-1,0} - 8u_{0,-1,0} + 10u_{-1,-1,0} \\
&\qquad - u_{-1,1,-1} - 4u_{0,0,-1} + 6u_{-1,0,-1} \\
&\qquad - u_{1,-1,-1} + 6u_{0,-1,-1} - 7u_{-1,-1,-1}) . \qquad (10)
\end{aligned}$$

This combination is a directional second order approximation to $D_1 = \partial u/\partial e_1$. While this approximation is different from the one given in equations 9, the relation between them is supplied by the terms Tx, Ty and Tz. The degrees of freedom described by the terms Tx, Ty and Tz in equation 9 correspond, among others, to the weights shown in figures 8-17. These figures are computer-generated images using the output from appendix 8 of the program described in the appendices 1-5. Except for a factor containing the grid spacings, each of these sets of weights is an approximation to a derivative, on a stencil which is the set of the points with a non-zero weight. In the present example, these are all approximations to combinations of various mixed fourth derivatives. Since these approximations appear only in the third order error term of the approximation to D_1, they can be added ad libitum without deteriorating the optimization of the second order approximation.

The derivative $(u_x)_{0,0,0}$ of equations 9 can be obtained by taking the derivative $(u_x)_{0,0,0}$ of equations 10 to which is added half the approximation of figure 9. The

derivative $(u_y)_{0,0,0}$ of equations 9 can be obtained by taking the derivative $(u_y)_{0,0,0}$ of equations 10 to which is added half the approximation of figure 11 and subtracted half the approximation of figure 9. The derivative $(u_z)_{0,0,0}$ of equations 9 can be obtained by taking the derivative $(u_z)_{0,0,0}$ of equations 10 to which is added half the approximation of figure 17 and subtracted half the approximation of figure 11.

Remark that in the present example the degrees of freedom translated into approximations which were invisible at the level of the leading error of the directional approximation. Also, the stencils related to the degrees of freedom are the same for each of the grid-based derivatives. This is not the case in general. Degrees of freedom normally result in approximations with the same order of accuracy as the leading error of the directional approximation. Here, it is the restrictive condition which is imposed on the error of the approximation which results in combinations of stencils of third derivatives which are effectively fourth derivatives. Also, for some optimizations for approximations to derivatives, the degrees of freedom in the different grid-based approximations are coupled. This means that e.g. Ty and Tz contain a coefficient which appears in both the u_y and u_z discretizations in such a way that their effect is annihilated for the specific directions along which the optimization takes place. It is also to be noted that adding an adirectional discretization of sufficiently high order to the approximation of the directional derivative does not harm the directional properties. Finally, taking into account degrees of freedom allowing larger stencils, e.g. larger than the domain indicated in figure 4 in the above example, effectively results in an unlimited number of degrees of freedom, and an unlimited number of discretizations which can be added.

The degrees of freedom which are represented by the terms Tx, Ty and Tz, some of which are illustrated in figures 8 to 17, offer an opportunity for further tuning of the approximation, especially in the case that the degrees of freedom correspond to approximations of sufficient low order. The use of an extended stencil with points off the coordinate lines open a vast area of optimization, e.g. in the Fourier domain for improved convergence. These newly found degrees of freedom are a crucial part of the invention. Other examples of the degrees of freedom are given further on.

As mentioned, the program given in the appendices is capable to find the terms Tx, Ty and Tz. When adding an elementary post-processing routine, it is possible to relate two different approximations using the different degrees of freedom, just as shown above between equations 9 and 10 .

The stencils indicated by the terms T represent a basis of discretizations. As with any basis, transformations to another basis is possible. Linear combinations of these stencils form another basis, and other bases can be expressed in the terms T. It is therefore to be understood that the terms T are just one representation, and that any transformation

to other stencils, linear combinations, orthogonalization, scaling, orthonormalization such as the Gramm-Schmidt orthonormalization and so on and so forth is part of the invention.

The above procedure describes the systematic construction of the discretizations of first derivatives (gradients) including degrees of freedom, which can be used directly in the Finite Difference and in the Finite Volume formulations, and which can be translated to other formulations. Some of the gradients are described below, but the scope of the invention is not limited to those.

A simulation method using directional discretizations for the first derivative $D_1 = \partial u/\partial e_1$ of order M and eliminating the terms $\partial^{M+1}u/\partial e_2^{M_2}\partial e_3^{M_3}\dots$ with $M_2 + M_3 + \dots = M + 1$.

The above method using a local basis $B(\vec{e}_1, \vec{e}_2, \vec{e}_3, \dots)$ of curvilinear coordinates which has the unit vector $\vec{e}_1$ along the preferential direction. The optimization is such that the terms $\partial^{M+1}u/\partial e_2^{M_2}\partial e_3^{M_3}\dots$ with $M_2 + M_3 + \dots = M+1$ are eliminated in the approximated value D_1^A.

When approximating the derivative $D_1 = \partial u/\partial e_1$ with a discretization which has an error of order M, the error term contains in general contributions $\partial^{M+1}u/\partial e_1^{M_1}\partial e_2^{M_2}\partial e_3^{M_3}\dots$ with $M_1+M_2+M_3+\dots = M+1$. It is advantageous to eliminate the error components which are normal to the preferential direction, which are the terms $\partial^{M+1}u/\partial e_2^{M_2}\partial e_3^{M_3}\dots$ with $M_2 + M_3 + \dots = M+1$. This is always possible with the proper choice of the terms T, where the terms T are taken constant or where they depend e.g. on the flow angles.

A simulation method using directional discretizations for the first derivative $D_1 = \partial u/\partial e_1$ of order M and eliminating the terms $\partial^{M+1}u/\partial e_1^{M_1}\partial e_2^{M_2}\partial e_3^{M_3}\dots$ with $M_1 + M_2 + M_3 + \dots = M + 1$ and $M_1 < M + 1$.

The above method using a local basis $B(\vec{e}_1, \vec{e}_2, \vec{e}_3, \dots)$ of curvilinear coordinates which has the unit vector $\vec{e}_1$ along the preferential direction. The optimization is such that the terms $\partial^{M+1}u/\partial e_1^{M_1}\partial e_2^{M_2}\partial e_3^{M_3}\dots$ with $M_1 + M_2 + M_3 + \dots = M+1$ and $M_1 < M+1$ are eliminated in the approximated value D_1^A in the case that $\vec{e}_1$ is along the x-axis or along diagonals.

This is an optimization of the approximation which reduces the discretization to a one-dimensional discretization along the x-axis and along diagonals. This discretization has a reduced error for preferential directions with $a \geq b \geq c \geq \dots \geq 0$, and permutations thereof for other directions, but the discretization can be applied to other preferential directions.

A simulation method using a second order directional discretization for the first derivative.

A simulation method which provides a simulated numerical representation which uses a second order directional discretization in N dimensions, $N \geq 3$, on a structured grid, for the approximation of D_1 derivatives with a reduced error is given by

$$\begin{pmatrix} (u_x)_{i,j,k,\ldots} \\ (u_y)_{i,j,k,\ldots} \\ (u_z)_{i,j,k,\ldots} \\ \vdots \end{pmatrix}, \tag{11}$$

wherein

$$\begin{aligned}
(u_x)_{i,j,k,\ldots} &= \frac{1}{\Delta x}\left\{\frac{1}{2}(u_{i+1,j,k,\ldots} - u_{i-1,j,k,\ldots}) + Tx\right\}, \\
(u_y)_{i,j,k,\ldots} &= \frac{1}{\Delta y}\left\{\frac{1}{2}(u_{i+1,j+1,k,\ldots} - u_{i+1,j,k,\ldots} + u_{i-1,j,k,\ldots} - u_{i-1,j-1,k,\ldots}) + Ty\right\}, \\
(u_z)_{i,j,k,\ldots} &= \frac{1}{\Delta z}\left\{\frac{1}{2}(u_{i+1,j+1,k+1,\ldots} - u_{i+1,j+1,k,\ldots} + u_{i-1,j-1,k,\ldots} - u_{i-1,j-1,k-1,\ldots}) + Tz\right\}, \\
&\vdots
\end{aligned} \tag{12}$$

This discretization is a special case of the discretization of equation 8, with $m = n = 1$, $a_1 = \frac{1}{2}$, $a_0 = 0$ and $a_{-1} = -\frac{1}{2}$. The terms Tx, Ty, Tz, $\ldots$ represent the degrees of freedom which are used in the optimization of said approximated value D_1^A, and where indices $(i, j, k, \ldots)$ define the point of computation P on the N-dimensional grid, and $\Delta x, \Delta y, \Delta z, \ldots$ denote the mesh spacings of the N-dimensional grid in each coordinate direction.

This discretization has a reduced error for preferential directions with $a \geq b \geq c \geq \ldots \geq 0$, and permutations thereof for other directions, but the discretization can be applied to other preferential directions.

An example in three dimensions has been discussed above.

A simulation method using a fourth order directional discretization for the first derivative.

A simulation method which provides a simulated numerical representation which uses a fourth order directional discretization for the approximation of D_1 derivatives in N dimensions, $N \geq 3$, on a structured grid with a reduced error is given by

$$\begin{pmatrix} (u_x)_{i,j,k,\ldots} \\ (u_y)_{i,j,k,\ldots} \\ (u_z)_{i,j,k,\ldots} \\ \vdots \end{pmatrix}, \tag{13}$$

wherein

$$(u_x)_{i,j,k,\ldots} = \frac{1}{\Delta x}\left\{\frac{1}{12}(u_{i-2,j,k,\ldots} - 8u_{i-1,j,k,\ldots} + 8u_{i+1,j,k,\ldots} - u_{i+2,j,k,\ldots}) + Tx\right\},$$

$$
\begin{aligned}
(u_y)_{i,j,k,\ldots} &= \frac{1}{\Delta y}\left\{\frac{1}{12}(u_{i-2,j-2,k,\ldots} - u_{i-2,j,k,\ldots}\right. \\
&\quad - 8u_{i-1,j-1,k,\ldots} + 8u_{i-1,j,k,\ldots} \\
&\quad + 8u_{i+1,j+1,k,\ldots} - 8u_{i+1,j,k,\ldots} \\
&\quad \left. - u_{i+2,j+2,k,\ldots} + u_{i+2,j,k,\ldots}) + Ty\right\}, \\
(u_z)_{i,j,k,\ldots} &= \frac{1}{\Delta z}\left\{\frac{1}{12}(u_{i-2,j-2,k-2,\ldots} - u_{i-2,j-2,k,\ldots}\right. \\
&\quad - 8u_{i-1,j-1,k-1,\ldots} + 8u_{i-1,j-1,k,\ldots} \\
&\quad + 8u_{i+1,j+1,k+1,\ldots} - 8u_{i+1,j+1,k,\ldots} \\
&\quad \left. - u_{i+2,j+2,k+2,\ldots} + u_{i+2,j+2,k,\ldots}) + Tz\right\}, \\
&\vdots
\end{aligned} \tag{14}
$$

This discretization is a special case of the discretization of equation 8, with $m = n = 2$, $a_{-2} = \frac{1}{12}$, $a_{-1} = -\frac{8}{12}$, $a_0 = 0$, $a_1 = \frac{8}{12}$ and $a_2 = -\frac{1}{12}$. The terms Tx, Ty, Tz, ... represent the degrees of freedom which are used in the optimization of said approximated value D_1^A, and where indices $(i, j, k, \ldots)$ define the point of computation P on the N-dimensional grid, and $\Delta x, \Delta y, \Delta z, \ldots$ denote the mesh spacings of the N-dimensional grid in each coordinate direction.

This discretization has a reduced error for preferential directions with $a \geq b \geq c \geq \ldots \geq 0$, and permutations thereof for other directions, but the discretization can be applied to other preferential directions.

A simulation method using a second order directional upwind discretization for the first derivative.

A simulation method which provides a simulated numerical representation which uses a second order directional upwind discretization for the approximation of D_1 derivatives in N dimensions, $N \geq 3$, on a structured grid with a reduced error is given by

$$
\begin{pmatrix} (u_x)_{i,j,k,\ldots} \\ (u_y)_{i,j,k,\ldots} \\ (u_z)_{i,j,k,\ldots} \\ \vdots \end{pmatrix}, \tag{15}
$$

wherein

$$
\begin{aligned}
(u_x)_{i,j,k,\ldots} &= \frac{1}{\Delta x}\left(\frac{3}{2}u_{i,j,k,\ldots} - 2u_{i-1,j,k,\ldots} + \frac{1}{2}u_{i-2,j,k,\ldots} + Tx\right), \\
(u_y)_{i,j,k,\ldots} &= \frac{1}{\Delta y}\Big(- 2(u_{i-1,j-1,k,\ldots} - u_{i-1,j,k,\ldots}) \\
&\qquad + \frac{1}{2}(u_{i-2,j-2,k,\ldots} - u_{i-2,j,k,\ldots}) + Ty\Big), \\
(u_z)_{i,j,k,\ldots} &= \frac{1}{\Delta z}\Big(- 2(u_{i-1,j-1,k-1,\ldots} - u_{i-1,j-1,k,\ldots})
\end{aligned}
$$

$$+ \frac{1}{2}(u_{i-2,j-2,k-2,\ldots} - u_{i-2,j-2,k,\ldots}) + Tz\Big) ,$$

$$\vdots \tag{16}$$

This discretization is a special case of the discretization of equation 8, with $m = 2$, $n = 0$, $a_0 = \frac{3}{2}$, $a_{-1} = -2$, and $a_{-2} = \frac{1}{2}$. The terms Tx, Ty, Tz, ... represent the degrees of freedom which are used in the optimization of said approximated value D_1^A, and where indices $(i, j, k, \ldots)$ define the point of computation P on the N-dimensional grid, and $\Delta x, \Delta y, \Delta z, \ldots$ denote the mesh spacings of the N-dimensional grid in each coordinate direction.

A simulation method using discretizations obtained by a computer program.

A simulation method which provides a simulated numerical representation in three dimensions wherein the approximation D_p^A with order M of the derivative $D_p = \partial^p u/\partial e_1^{p_1} \partial e_2^{p_2} \partial e_3^{p_3}$ with $p_1 + p_2 + p_3 = p$ on a grid of given extent is obtained from the output of the program generate-discretizations which is given in appendices 1-5.

This program has been mentioned before. The program generates the grid-based approximations used in the discretization of the derivative $D_p = \partial^p u/\partial e_1^{p_1} \partial e_2^{p_2} \partial e_3^{p_3}$ with $p_1 + p_2 + p_3 = p$ with order M on a grid of given extent. An optimization parameter `optimize` controls which type of optimization of D_p^A is generated.

A description of the input parameters and its use can be found in the listings in appendices 1-5.

For `optimize=0`, the program generates the basis of stencils resulting in a consistent approximation of D_p^A. The user can then use this basis to optimize D_p^A as desired. Any of the discretizations of the invention in three dimension in the Finite Difference formulation can be expressed in this basis, and therefore, any basis obtained with `optimize=0` represents the essence of the invention. The extension to other formulations or to more dimensions is trivial.

The rotations which have been used in the axes transformation, equation 6, are the following. It takes two successive rotations to link the e_1-axis of the local basis B with the fixed axes. The first is in the grid basis G around the z-axis with angle α, resulting in the axis system $A'(x', y', z')$ with $z' = z$. This is followed by a rotation around the y'-axis with angle β, as shown in figure 18. The axis system is $A''(x'', y'', z'')$, with $y'' = y'$. Another rotation with angle γ around the $x'' = e_1$-axis positions the e_2 and e_3 axes, giving the directional axis system of the local basis B. The rotations are given by

their matrices, $\vec{x} = M_\alpha \vec{x}'$, $\vec{x}' = M_\beta \vec{x}''$ and $\vec{x}'' = M_\gamma \vec{e}$,

$$M_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad M_\beta = \begin{pmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{pmatrix},$$

$$M_\gamma = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix}, \tag{17}$$

and the transformation $\vec{x} = M\vec{e} = M_\alpha M_\beta M_\gamma \vec{e}$ is defined by

$$M = \begin{pmatrix} \cos\alpha\cos\beta & -\sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma & -\sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma \\ \sin\alpha\cos\beta & \cos\alpha\cos\gamma + \sin\alpha\sin\beta\sin\gamma & \cos\alpha\sin\gamma - \sin\alpha\sin\beta\cos\gamma \\ \sin\beta & -\cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix}. \tag{18}$$

The inverse of M, $M^{-1} = M^T$, is used in $\vec{e} = M^{-1}\vec{x}$, and in the expression for the derivative along e_1,

$$\frac{\partial u}{\partial e_1} = \cos\alpha\cos\beta\frac{\partial u}{\partial x} + \sin\alpha\cos\beta\frac{\partial u}{\partial y} + \sin\beta\frac{\partial u}{\partial z}, \tag{19}$$

a transformation which only involves the angles α and β. In the listing of the program in the appendices, the choice $\gamma = 0$ is coded and active, but the more general choice of γ is also coded, and can easily be activated by uncommenting the appropriate lines of code.

The above choice of rotations has the advantage of resulting in a very simple transformation. Of course, other choices of rotations are possible, and the above is not meant to exclude any of the other transformations from the invention.

With the above choice of transformation, in three dimensions, the continuity of the approximation with respect to changing flow directions imposes the following symmetries :

1. Flow in the $z = 0$ plane, $\beta = 0$. Then, ${}^{\partial u}/_{\partial e_1} = \cos\alpha {}^{\partial u}/_{\partial x} + \sin\alpha {}^{\partial u}/_{\partial y}$. The stencil should have mirror symmetry with respect to the $z = 0$ plane.

2. Flow in the $x = y$ plane, $\alpha = {}^{\pi}/_{4}$ and ${}^{\partial u}/_{\partial e_1} = {}^{1}/_{2}\sqrt{(2)}\cos\beta({}^{\partial u}/_{\partial x} + {}^{\partial u}/_{\partial y}) + \sin\beta {}^{\partial u}/_{\partial z}$. The stencil should have mirror symmetry with respect to the plane $x = y$.

3. Flow in the $y = z$ plane, $\beta = \arctan(\sin(\alpha))$ with ${}^{\partial u}/_{\partial e_1} = (1 + \sin^2\alpha)^{-1/2}$ $[\cos\alpha {}^{\partial u}/_{\partial x} + \sin\alpha({}^{\partial u}/_{\partial y} + {}^{\partial u}/_{\partial z})]$. The stencil should have mirror symmetry with respect to the plane $y = z$.

4. Flow along the x-axis, $\alpha = \beta = 0$. Conditions 1 and 3 apply.

5. Flow along the diagonal in the x-y plane, $\alpha = \pi/4$, $\beta = 0$. Conditions 1 and 2 apply.

6. Flow along the body diagonal in x-y-z space, $\alpha = \pi/4$, $\beta = \beta^*$ where $\beta^* = \arctan(1/2\sqrt{(2)})$. Conditions 2 and 3 apply.

In the following, a simulation method is presented, using a discretization obtained from the program described in the appendices with certain input parameters. While the examples have practical importance, they do not mean to restrict the invention, and do not intend to exclude from the invention any other combination of input parameters.

A simulation method using discretizations obtained by the computer program with the input variable `optimize=1`

For `optimize=1` and for a first derivative, the program generates the output just mentioned, together with the conditions which are sufficient to eliminate the terms $\partial^{M+1}u/\partial e_2^{M_2}\partial e_3^{M_3}$ with $M_2 + M_3 = M + 1$. This means that an additional $M + 2$ conditions are imposed. This is an optimization discussed before.

A simulation method using discretizations obtained by the computer program with the input variable `optimize=2`

For `optimize=2`, the program generates the basis of stencils resulting in a consistent approximation of D_p^A eliminating the terms $\partial^{M+1}u/\partial e_1^{M_1}\partial e_2^{M_2}\partial e_3^{M_3}$ with $M_1 + M_2 + M_3 = M + 1$ and $M_1 < M + 1$ in the case that $\vec{e}_1$ is along the x-axis or along diagonals. This optimization reduces the number of degrees of freedom. This is an optimization discussed before.

A simulation method using discretizations obtained by the computer program with the input variable `order=1`

For `order=1`, the program generates the basis of stencils resulting in a consistent first order approximation of D_p^A. First order accurate discretizations can be monotone, and are important in simulations with strong gradients. Higher order discretizations will give overshoots in the approximated values. Commonly, a combination of limiting functions or limiters is used to switch the approximation near discontinuities to first order monotone discretizations. This reduces locally the order of the approximation but preserves monotonicity of the solution.

A simulation method using discretizations obtained by the computer program with the input `gridmin=-1, gridmax=1, se1=1, se2=0, se3=0, order=2, optimize=0, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program

generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-1, gridmax=1,`
2. `se1=1, se2=0, se3=0,`
3. `order=2, optimize=0, check=0.`

The output obtained with this input can be found in appendix 6. Remark that the 17 stencils which are associated with the degrees of freedom are the same for u_x, u_y and u_z.

A simulation method using discretizations obtained by the computer program with the input `gridmin=-1, gridmax=1, se1=1, se2=0, se3=0, order=2, optimize=1, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-1, gridmax=1,`
2. `se1=1, se2=0, se3=0,`
3. `order=2, optimize=1, check=0.`

The output obtained with this input can be found in appendix 7. Remark that the stencils which are associated with the degrees of freedom are the same for u_x, u_y and u_z. The difference with the output for `optimize=1` is the four extra conditions at the end of the output.

A simulation method using discretizations obtained by the computer program with the input `gridmin=-1, gridmax=1, se1=1, se2=0, se3=0, order=2, optimize=2, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-1, gridmax=1,`
2. `se1=1, se2=0, se3=0,`
3. `order=2, optimize=2, check=0.`

The output obtained with this input can be found in appendix 8. Remark that the stencils which are associated with the degrees of freedom are the same for u_x, u_y and u_z,

but differ from the stencils obtained with the output for `optimize=0`. The optimization has reduced the number of degrees of freedom from 17 to 10.

A simulation method using discretizations obtained by the computer program with the input `gridmin=-2, gridmax=2, se1=1, se2=0, se3=0, order=4, optimize=0, check=0`
A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-2, gridmax=2,`
2. `se1=1, se2=0, se3=0,`
3. `order=4, optimize=0, check=0.`

A part of the output obtained with this input can be found in appendix 9. Since the stencils which are associated with the degrees of freedom are the same for u_x, u_y and u_z, only those for u_x are listed. The cuts are indicated in the output.

A simulation method using discretizations obtained by the computer program with the input `gridmin=-2, gridmax=2, se1=1, se2=0, se3=0, order=4, optimize=1, check=0`
A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-2, gridmax=2,`
2. `se1=1, se2=0, se3=0,`
3. `order=4, optimize=1, check=0.`

A part of the output obtained with this input can be found in appendix 10. The 90 stencils which are associated with the degrees of freedom are the same for `optimize=0` and are omitted. The six conditions would take too much space to print, so just the beginning of the conditions is shown.

A simulation method using discretizations obtained by the computer program with the input `gridmin=-2, gridmax=2, se1=1, se2=0, se3=0, order=4, optimize=2, check=0`
A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-2, gridmax=2,`

2. `se1=1, se2=0, se3=0,`

3. `order=4, optimize=2, check=0.`

A part of the output obtained with this input can be found in appendix 11. Since the stencils which are associated with the degrees of freedom are the same for u_x, u_y and u_z, only those for u_x are listed. The cuts are indicated in the output. We have now 72 degrees of freedom for each grid-based derivative in the basis of stencils.

Given the enormous amount of output for larger stencils, we omit from now on to give the output.

A simulation method using discretizations obtained by the computer program with the input `gridmin=-2, gridmax=2, se1=1, se2=0, se3=0, order=3, optimize=0, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-2, gridmax=2,`

2. `se1=1, se2=0, se3=0,`

3. `order=3, optimize=0, check=0.`

There are now 105 degrees of freedom for each grid-based derivative in the basis of stencils.

A simulation method using discretizations obtained by the computer program with the input `gridmin=-2, gridmax=2, se1=1, se2=0, se3=0, order=3, optimize=1, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-2, gridmax=2,`

2. `se1=1, se2=0, se3=0,`

3. `order=3, optimize=1, check=0.`

A simulation method using discretizations obtained by the computer program with the input `gridmin=-2, gridmax=2, se1=1, se2=0, se3=0, order=3, optimize=2, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-2, gridmax=2,`
2. `se1=1, se2=0, se3=0,`
3. `order=3, optimize=2, check=0.`

There are now 92 degrees of freedom for each grid-based derivative in the basis of stencils.

A simulation method using discretizations obtained by the computer program with the input `gridmin=-3, gridmax=3, se1=1, se2=0, se3=0, order=6, optimize=0, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-3, gridmax=3,`
2. `se1=1, se2=0, se3=0,`
3. `order=6, optimize=0, check=0.`

There are now 259 degrees of freedom for each grid-based derivative in the basis of stencils.

A simulation method using discretizations obtained by the computer program with the input `gridmin=-3, gridmax=3, se1=1, se2=0, se3=0, order=6, optimize=1, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-3, gridmax=3,`
2. `se1=1, se2=0, se3=0,`
3. `order=6, optimize=1, check=0.`

A simulation method using discretizations obtained by the computer program with the input `gridmin=-3, gridmax=3, se1=1, se2=0, se3=0, order=6, optimize=2, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-3, gridmax=3,`
2. `se1=1, se2=0, se3=0,`
3. `order=6, optimize=2, check=0.`

A simulation method using discretizations obtained by the computer program with the input `gridmin=-3, gridmax=3, se1=1, se2=0, se3=0, order=5, optimize=0, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-3, gridmax=3,`
2. `se1=1, se2=0, se3=0,`
3. `order=5, optimize=0, check=0.`

There are now 287 degrees of freedom for each grid-based derivative in the basis of stencils.

A simulation method using discretizations obtained by the computer program with the input `gridmin=-3, gridmax=3, se1=1, se2=0, se3=0, order=5, optimize=1, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-3, gridmax=3,`
2. `se1=1, se2=0, se3=0,`
3. `order=5, optimize=1, check=0.`

A simulation method using discretizations obtained by the computer program with the input `gridmin=-3, gridmax=3, se1=1, se2=0, se3=0, order=5, optimize=2, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-3, gridmax=3,`

2. `se1=1, se2=0, se3=0,`

3. `order=5, optimize=2, check=0.`

A simulation method using discretizations obtained by the computer program with the input `gridmin=-1, gridmax=1, se1=2, se2=0, se3=0, order=2, optimize=0, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-1, gridmax=1,`

2. `se1=2, se2=0, se3=0,`

3. `order=2, optimize=0, check=0.`

The output of the program with the above input parameters is given in appendix 12.

A simulation method using discretizations obtained by the computer program with the input `gridmin=-1, gridmax=1, se1=2, se2=0, se3=0, order=2, optimize=2, check=0`

A simulation method in three dimensions which uses the program described in the appendices 1-5, wherein the approximation D_p^A is obtained from the output of the program generate-discretizations which is given in appendices 1-5 using the input parameters

1. `gridmin=-1, gridmax=1,`

2. `se1=2, se2=0, se3=0,`

3. `order=2, optimize=2, check=0.`

A simulation method using a second order directional discretization for the second derivative in three dimensions.

A simulation method which provides a simulated numerical representation which uses a second order directional discretization in three dimensions for the second derivative $D_2 = \partial^2 u/\partial e_1^2$, which is expressed in the terms u_{xx}, u_{yy}, u_{zz}, u_{xy}, u_{yz} and u_{zx} which are given by

$$
\begin{aligned}
(u_{xx})_{i,j,k} &= \frac{1}{(\Delta x)^2}\left(u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k} + Txx\right) , \\
(u_{yy})_{i,j,k} &= \frac{1}{(\Delta y)^2}\left(u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k} + Tyy\right) ,
\end{aligned}
$$

$$
\begin{aligned}
(u_{zz})_{i,j,k} &= \frac{1}{(\Delta z)^2}\left(u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1} + Tzz\right) , \\
(u_{xy})_{i,j,k} &= \frac{1}{\Delta x \Delta y}\left\{\frac{1}{2}(u_{i+1,j+1,k} - u_{i,j+1,k} - u_{i+1,j,k} + 2u_{i,j,k}\right. \\
&\qquad \left. -u_{i-1,j,k} - u_{i,j-1,k} + u_{i-1,j-1,k}) + Txy\right\} , \\
(u_{yz})_{i,j,k} &= \frac{1}{\Delta y \Delta z}\left\{\frac{1}{4}(u_{i+1,j+1,k+1} - u_{i+1,j,k+1} - u_{i+1,j+1,k} + u_{i+1,j,k}+\right. \\
&\qquad u_{i,j+1,k+1} - u_{i,j,k+1} - u_{i,j+1,k} + 2u_{i,j,k} \\
&\qquad -u_{i,j-1,k} - u_{i,j,k-1} + u_{i,j-1,k-1}+ \\
&\qquad \left. u_{i-1,j-1,k-1} - u_{i-1,j-1,k} - u_{i-1,j,k-1} + u_{i-1,j,k}) + Tyz\right\} , \\
(u_{zx})_{i,j,k} &= \frac{1}{\Delta z \Delta x}\left\{\frac{1}{4}(u_{i+1,j+1,k+1} - u_{i,j+1,k+1} - u_{i+1,j+1,k} + u_{i,j+1,k}+\right. \\
&\qquad u_{i+1,j,k+1} - u_{i+1,j,k} - u_{i,j,k+1} + 2u_{i,j,k} \\
&\qquad -u_{i-1,j,k} - u_{i,j,k-1} + u_{i-1,j,k-1}+ \\
&\qquad \left. u_{i-1,j-1,k-1} - u_{i-1,j-1,k} - u_{i,j-1,k-1} + u_{i,j-1,k}) + Tzx\right\} , \qquad (20)
\end{aligned}
$$

wherein the terms Txx, Txy, Txz, Tyy, Tyz and Tzz represent the degrees of freedom which are used in the optimization of said approximated value D_2^A, and where indices (i, j, k) define the point of computation P on the three-dimensional grid, and Δx, Δy, Δz denote the mesh spacings of the three-dimensional grid in each coordinate direction.

The second derivative $D_2 = \partial^2 u/\partial e_1^{2_1}$ can be expressed in the grid-based derivatives u_{xx}, u_{yy}, u_{zz}, u_{xy}, u_{yz} and u_{zx} according to

$$
\begin{aligned}
\frac{\partial^2 u}{\partial e_1^2} = {}& \cos^2\alpha \cos^2\beta \frac{\partial^2 u}{\partial x^2} + \sin^2\alpha \cos^2\beta \frac{\partial^2 u}{\partial y^2} + \sin^2\beta \frac{\partial^2 u}{\partial z^2} + \\
& 2\cos\alpha \cos^2\beta \sin\alpha \frac{\partial^2 u}{\partial x \partial y} + 2\cos\alpha \cos\beta \sin\beta \frac{\partial^2 u}{\partial x \partial z} + 2\sin\alpha \cos\beta \sin\beta \frac{\partial^2 u}{\partial y \partial z} . \qquad (21)
\end{aligned}
$$

The terms Txx, Txy, Txz, Tyy, Tyz and Tzz are obtained by the computer program mentioned before, and are given in appendix 12. The stencils Txx, Txy, Txz, Tyy, Tyz and Tzz have been added in various quantities to the stencils Tf_{xx}, Tf_{xy}, Tf_{xz}, Tf_{yy}, Tf_{yz} and Tf_{zz} to obtain the more symmetric representation of equation 20. This shows once more the use of the degrees of freedom, and the equivalence between two expressions for an approximation of D_p using a different Tf but sharing the basis described by the stencils.

A simulation method using the second order directional central discretization for the first derivative in the Finite Element formulation.

A second order directional central discretization in 3 dimensions on a structured grid, for the approximation of the D_1 derivative $D_1 = \partial u/\partial e_1$ with a reduced error when the preferential direction $\vec{a}$ is directed along certain diagonals of the grid, is constructed as follows.

For the division of space in elements, we use the subdivision of a cube in tetrahedra, as indicated in figure 19. This subdivision holds for all the cubes of the part of space under consideration. The element for point P is defined as the conjunction of all tetrahedra meeting in P. For the basis functions ϕ the standard linear basis functions for tetrahedra, as described in *D1*, are taken. The weighting function ψ is defined as : $\psi = 0$ over the entire element, except for the tetrahedra A and B, where $\psi = 3$. Tetrahedron A is defined in the cube where P is situated at vertex 1 as the tetrahedron formed by the vertices 1, 2, 3 and 7. Tetrahedron B is defined in the cube where P is situated at vertex 7 as the tetrahedron formed by the vertices 7, 8, 5 and 1 (see figure 19).

The resulting discretization is the second order directional central discretization for the first derivative.

With different elements, basis functions and test functions, other directional discretizations can be formed, as anyone skilled in numerical methods is able to verify.

A simulation method using combined discretizations.

Some discretizations are commonly expressed as a sum of separate discretizations. An example is the combination of the second order central discretization and an upwind discretization,

$$\begin{aligned}
(u_x)_{i,j,k,\ldots} &= \frac{1}{2\Delta x}\left(u_{i+1,j,k,\ldots} - u_{i-1,j,k,\ldots}\right) \\
&\quad -\frac{1}{6\Delta x}\left(u_{i+1,j,k,\ldots} - 3u_{i,j,k,\ldots} + 3u_{i-1,j,k,\ldots} - u_{i-2,j,k,\ldots}\right), \\
(u_y)_{i,j,k,\ldots} &= \frac{1}{2\Delta y}(u_{i+1,j+1,k,\ldots} - u_{i+1,j,k,\ldots} + u_{i-1,j,k,\ldots} - u_{i-1,j-1,k,\ldots}) \\
&\quad + \frac{1}{6\Delta y}(-u_{i-2,j-2,k,\ldots} + u_{i-2,j,k,\ldots} + 3u_{i-1,j-1,k,\ldots} \\
&\qquad -3u_{i-1,j,k,\ldots} - u_{i+1,j+1,k,\ldots} + u_{i+1,j,k,\ldots}), \\
(u_z)_{i,j,k,\ldots} &= \frac{1}{2\Delta z}(u_{i+1,j+1,k+1,\ldots} - u_{i+1,j+1,k,\ldots} + u_{i-1,j-1,k,\ldots} - u_{i-1,j-1,k-1,\ldots}) \\
&\quad + \frac{1}{6\Delta z}(-u_{i-2,j-2,k-2,\ldots} + u_{i-2,j-2,k,\ldots} + 3u_{i-1,j-1,k-1,\ldots} \\
&\qquad -3u_{i-1,j-1,k,\ldots} - u_{i+1,j+1,k+1,\ldots} + u_{i+1,j+1,k,\ldots}), \\
&\;\;\vdots
\end{aligned} \tag{22}$$

where for simplicity of notation the terms Tx, Ty, Tz, ... are omitted.

It will be clear that combined discretizations have their directional counterparts, which for the first derivatives in the most general case are obtained with equation 8. In this respect, the invention thus concerns a method wherein $D_p^A = \sum_n L_n D_{p,n}^A$ where L_n are constants and each $D_{p,n}^A$ is a function of value $u_{\hat{s}}$ of the parameter in a collection of grid points, called stencil S_n, with individual computational functions, which depend on the numerical framework in which $D_{p,n}^A$ is expressed. In the approximation $D_{p,n}^A$, the

computational functions are chosen in such a way that the approximated value $D^A_{p,n}$ is optimized for the preferential direction.

A simulation method using combined discretizations with limiters. A special case of combined schemes is the use of limiting functions, called limiters, which are at present a common practice. These limiters are used to change locally the discretization, by using a blending function between a higher-order discretization, and a low-order one, typically a first order scheme. The purpose is to obtain a discretization which does not violate a monotonicity principle.

Limiting functions can be applied to schemes using directional discretizations, thereby blending different directional discretizations.

In this respect, the invention thus concerns a method wherein $D^A_p = \sum_n L_n D^A_{p,n}$ where L_n are limiting functions of the values u_s of the stencil S, and at least one $D^A_{p,n}$ is a function of value u_s of the parameter in a collection of grid points, called stencil S_n, with individual computational functions, which depend on the numerical framework in which $D^A_{p,n}$ is expressed. In the approximation $D^A_{p,n}$, the computational functions are chosen in such a way that the approximated value $D^A_{p,n}$ is optimized for the preferential direction.

A simulation method using upwind and central combined discretizations. Another general way to view combined discretizations is to consider the class of combinations which are chosen from the group of : upwind discretizations, centered discretizations, or discretizations which are a combination of at least one upwind discretization and at least one centered discretization.

A simulation method using nonlinear discretizations. Most of the physical phenomena which are encountered in real world applications are non-linear. An example is the non-linear advection equation

$$u_t + \vec{\nabla}\vec{f} = 0\,, \tag{23}$$

where the flux $\vec{f}(u)$ is a non-linear function of u in N dimensions. However, a local advection speed can be defined in many ways, e.g for the k^{th} component of $\vec{a}$, $a_k = \partial f_k/\partial u$, or $\vec{a}$ such that $\vec{\nabla}\cdot\vec{f} = \vec{a}\cdot\vec{\nabla}u$. Various options are available for the discretization of equation 23, either by using directly a space discretization of $\vec{f}$ where the stencil depends on $\vec{a}$, or by linearizing the equation in one form or another, leading to a space derivative of u. Various examples can be found in the books of Hirsch (*D1*). In any case, the directional discretizations can be used for any of the space derivatives, in order to reduce the discretization error and to simulate any space derivative of u according to the invention.

A simulation method with numerical schemes using several discretizations.

One step further then just combining discretizations in a joint discretization, is the use of different discretizations in a numerical scheme. A good example is the Lax-Wendroff scheme, see (*D6*) P.D. Lax and B. Wendroff, "Systems of conservation laws", Comm. Pure and Appl. Math., **13** (1960), pp. 217-317. This numerical scheme is based on a Taylor development in time for the unknown at point of computation P,

$$u^{n+1} = u^n + \Delta t u_t + \frac{1}{2}\Delta t^2 u_{tt} + \dots \,. \tag{24}$$

When the Lax-Wendroff scheme is used to simulate the solution of an N-dimensional advection equation,

$$u_t + \vec{a}.\vec{\nabla} u = 0\,, \tag{25}$$

the partial derivatives with respect on time in equation 24 can be rewritten in space derivatives. A simplification can be immediately applied, since $\vec{a}.\vec{\nabla}u = a_{e_1}u_{e_1} = a_{e_1}\partial u/\partial e_1$, that is the first derivative along $\vec{e}_1$ with velocity a_{e_1}. When the Taylor series is truncated after the second term, the resulting scheme is

$$u^{n+1} = u^n - a_{e_1}\Delta t u_{e_1} + \frac{1}{2}(a_{e_1}\Delta t)^2 u_{e_1e_1}\,. \tag{26}$$

The space derivatives can again be approximated by the directional discretizations. The first derivative is commonly discretized with a central discretization. For this we can use equation 12, or any other convenient directional discretization. The higher order terms can also be taken care of with directional discretizations.

Other schemes are the Lax-Friedrich scheme, and the scheme of Beam and Warming, which contain discretizations for which the invention is particularly appropriate.

The above examples have in common that a combination of various terms is used for certain numerical purposes, of which the enhancement of stability and the improvement of accuracy are the most common and important ones. This involves additional derivatives, each of which can be approximated by a directional discretization. The list of those schemes is long, and contains among others the Lax-Wendroff scheme, the Lax-Friedrich scheme, the MacCormack scheme, the leap-frog scheme, the Crank-Nicholson scheme, the Stone-Brian scheme, the box scheme, Henn's scheme, the QUICK scheme, the κ scheme, the Flux Corrected Transport (FTC) scheme, the family of ENO schemes, schemes in the class of the Piecewise Parabolic Method (PPM), multi-level schemes, schemes obtained with the fractional step method, and so on and so forth. It is to be understood that variations of these schemes which are related to the above schemes are included. The invention applies in general to every of the available schemes. An extended survey of numerical schemes can be found in the current literature as covered by the journals cited before, or in reference works like e.g *D1*, and references therein.

The purpose of the directional discretizations is the use in numerical schemes for the simulation of physical processes. The above schemes are some examples, but the application of the directional discretizations is not limited to those.

A simulation method using compact discretizations.

Among the numerical discretizations striving for accuracy is the class of compact discretizations. Compact discretizations generally store on the grid not only the variables u, but also first and higher derivatives of u, which are considered as independent variables. This leads to discretizations which cover a very compact stencil. However, the derivatives appearing in this formulation still profit from a directional approach, and the discretizations mentioned above can be applied to compact discretizations.

This is an example of a numerical discretization where the invention can be applied to at least one p^{th} discretization.

A simulation method for the discretization of systems of equations.

Many physical phenomena concern interrelated parameters. An example is the flow of a gas, which involves the coupled development of, among other, energy, density and velocity. Such processes are described by coupled systems of equations, where the mathematical description contains space derivatives, and where the description is completed by one or more equations of state. The discretization of systems of equations is well-developed, and the directional discretizations of the invention offer advantages for problems modeled by systems of equations such as the Navier-Stokes equations, the Euler equations, the shallow water equations, magneto-hydrodynamic equations and so on. For a description of the application of discretizations to systems of equations, see the paper *D2* or the books *D1*, which also include a description of the hierarchy of equations.

Application to the Navier-Stokes equations.

A major application of the use of numerical tools is in the simulation of physical phenomena which can be modeled, at least in part, by the Navier-Stokes equations. Closure of the equations is obtained by the use of equations(s) of state which depend on the physics considered. The Navier-Stokes equations are fundamental equations in industrial applications, such as flow simulations for vehicles, turbines, oil flow (including multiphase phenomena for oil recovery), ..., and they take a central position in the books *D1*. These equations are characterized by the fact that they form a coupled system of equations, which contain non-linear terms, both a first derivative and a second derivative. When the Navier-Stokes equations are simplified, other systems of equations result, which are special instants of the Navier-Stokes equations. Examples are the Euler equations, thin-layer approximations, incompressible Navier-Stokes equations and so on.

Application to the Euler equations.

A major application of numerical tools is the simulation of physical phenomena which

can be modeled, at least in part, by the Euler equations. This is a special case of the Navier-Stokes equations, and the equations are characterized by the fact that they form a coupled system of equations, which contain non-linear terms, and only first derivatives.

Application to the magneto-hydrodynamic equations.

Another major application of numerical tools is the simulation of physical phenomena which can be described by the coupling of the hydrodynamic equations with the electromagnetic equations. The directional discretizations of this invention are very well suited for application to equations with the combination of these characteristics.

Combinations of models.

Commonly, physical phenomena involve many parts of physics, such as fluid mechanics possibly involving multiple phases, electromagnetism, chemical reactions, stress analysis, heat transfer and so on. An example is the design and optimizations of a wing in a flow, taking into account the flexibility of the wing. Such combinations of physical phenomena are commonly described by the term multi-physics. The mathematical description of such combined physical phenomena involves therefore multiple sets of equations, among which equations which model turbulence, chemical reactions, electromagnetic phenomena, multiphase flow and multi-physics phenomena.

Directional discretizations combined with acceleration techniques.

The solution of the discrete equations can be accelerated by a number of well-known techniques, see e.g. the books *D1*. Among them are local time-stepping, preconditioning, multi-grid and GMRES. The directional discretizations of this invention are very well suited for application to discretizations with the combination of acceleration techniques.

Simulation of flow.

In the beginning of this description, the aim of the invention was stated as aiming at improving the numerical tool for the design and the optimization of commercial products, thereby reducing the computational cost of numerical simulations and the time of the design-cycle. A major application concerns the simulation of physical phenomena which include flow.

Simulation of an object in flow.

Numerical simulations are a valuable tool in the design and optimization of objects which are exposed to a flow. In particular, the invention aims to be used for obtaining a simulated numerical representation of a material object interacting with material flow.

Simulation of a vehicle.

A particular case is the design and optimization of vehicles which are moving in a flow. Vehicles can be used at all kind of velocities in all kind of flows. Examples are spacecraft in rarefied flow at hypersonic speeds, supersonic, transonic and subsonic aircraft, vehicles moving on land from racing cars to low-speed energy saving designs, underwater vehicles

like submarines, multiphase vehicles like boats, and so on and so forth. In particular, the invention aims to be used for obtaining a simulated numerical representation including a vehicle.

Simulation of rotating blades.

A particular case is the design and optimization of turbo-machines. They are member of the class of machines which contain a rotating part exposed to flow. This class includes also propellers as found on aircraft and ships, wind mills, ventilators, air conditioning systems, and so on. In particular, the invention aims to be used for obtaining a simulated numerical representation including a rotating blade.

Simulation of atmospheric flow.

A particular case of flow is the flow of the atmosphere. Simulation of this flow is at the basis of weather prediction. In particular, the invention aims to be used for obtaining a simulated numerical representation of the atmosphere.

Simulation of oil recovery.

A special case of multi-phase flow is the field of oil recovery, where oil is extracted by the use of a different fluid such as water or steam. In particular, the invention aims to be used for obtaining a simulated numerical representation applied to oil recovery.

Simulation of combustion.

A special case of multi-phase flow is combustion. This can be free combustion such as in forest fire, or internal combustion such as in an engine. In particular, the invention aims to be used for obtaining a simulated numerical representation of combustion.

Simulation using a computer

The tool of preference for numerical simulations is a computer. In particular, the invention aims to be used in a data processing system programmed to implement a simulation method according to the invention, so as to provide a simulated representation of said phenomenon from a data set of initial values of each of said parameter(s) u.

Simulation computer program.

A computer needs a computer program to execute the simulation. In particular, the invention aims to be used in a computer program that can be loaded in a data processing system so as to implement a method according to the invention, and to provide a simulated representation of said phenomenon from a data set of initial values of each of said parameter(s) u.

Simulation stored on a medium.

A computer program according to the invention needs to be stored. In particular, the invention aims to be used in a digital storage computer-readable medium containing a stored computer program that is configured to be loadable in a data processing system so as to implement a method according to the invention, and to provide a simulated

representation of said phenomenon from a data set of initial values of each of said parameter(s) u.

Applications in the framework of established numerical methods.

The methods of numerical simulations are well-developed, see e.g. *D1*. The invention fits smoothly into this framework, since it touches the core, namely the discretization of space derivatives, but without touching the established methods like acceleration techniques, preconditioners, grid adaptation and so on. It is to be understood that the invention applies to the entire established collected wisdom in the field of numerical simulations.

Although the invention is presented in its elementary form, any person competent in numerical methods can extend the presented discretizations for use on any type of grid including grids in polar, spherical or any type of coordinates, in non-linear applications, re-formulation in any of the formulations such as Finite Element, Finite Volume, or distribution formulations such as Residual Distribution or Flux Vector Distribution or any other formulation, derive the discretization for any directional first or higher derivative of any order, and any combination of the discretizations of the present invention in any numerical scheme like the Lax-Wendroff type of schemes, compact schemes of any type, and application to systems of equations.

The principle of the invention has been described. It will be clear to the experienced in numerical methods that various changes may be made to the embodiments described above without departing from the spirit and scope of this invention as described in the following claims. It is, therefore, to be understood that this invention is not limited to the specific details shown and described.

List of symbols

A	an amplitude of a Fourier component
A', A''	intermediate bases in the transformation from the grid basis to the local basis B
$a, b, c, \ldots$	components of the vector $\vec{a}$
$\vec{a}$	vector of preferential direction
$B(\vec{e}_1, \vec{e}_2, \vec{e}_3, \ldots)$	a local basis, with $\vec{e}_1$ along a preferential direction i.e. $\vec{e}_1 // \vec{a}$
C_c	computational coefficients used in the approximation of D_p^A, which are dependent on the numerical formulation used
C_s	computational coefficients used in the approximation of D_p^A, in the Finite Difference formulation ; weighting coefficients

$C_{l,m,n,\ldots}$	the weighting coefficients C_s for node $l, m, n, \ldots$
D_p	spatial p^{th} derivative, to be discretized
D_p^A	an approximation to D_p
D_p^{LC}	an approximation to D_p in the Finite Difference formulation, representing a linear combination of values
$D_p^{\alpha_i}$	an approximation to D_p in a Distribution Method, depending on the distribution coefficients α_i
$D_p^{\phi,\psi}$	an approximation to D_p in the Finite Element formulation, depending on the basis function ϕ and the test function ψ
f	the flux
I	the imaginary unit, such that $I^2 = -1$
$i, j, k, \ldots$	indices numbering the nodes of a structured grid
$i_{\max}, j_{\max}, k_{\max}, \ldots$	maximum indices of a grid
I_{el}	the integral of the derivative D_p over a volume, used in the Residual Distribution Method
M	order of the error of a discretization
N	number of dimensions
P	the point where the derivative is computed
p	index for a higher (p^{th}) derivative, or a first derivative ($p = 1$)
$p_1, p_2, p_3, \ldots$	the powers of the derivatives with respect to $\vec{e}_1$, $\vec{e}_2$, ... in a mixed derivative
$q_1, q_2, q_3, \ldots$	arbitrary variables
r	an integer summation index
$r_{\max}$	the maximum value of r in the summation
S	the stencil : the set of points used in the computation of the approximation D_p^A
t	the time coordinate
$t_{11}, t_{12}, \ldots$	coefficients used in the transformation between coordinate systems
$T\beta$	represent the terms in the discretization β resulting from degrees of freedom which remain when the approximated value D_p^A is optimized

u	unknown at a grid point
u_s	unknown at a point of the stencil S
u_α	derivative of u with respect to α, $u_\alpha = \frac{\partial u}{\partial \alpha}$
$\vec{x}$	N-dimensional position vector
$\vec{\nabla}$	differential operator, $\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}, \ldots\right)^T$
α_i	the distribution coefficient used in the Distribution Methods for the distribution of the part $\alpha_i I_{el}$ or $\alpha_i f$ to node i
$\Delta x, \Delta y, \Delta z, \ldots$	the mesh spacings in the coordinate directions
Δt	the time increment
$\frac{\partial u}{\partial x}$	partial derivative of u with respect to x
$\frac{\partial^p u}{\partial x^p}$	partial p^{th} derivative with of u respect to x
ϵ_n	error term in D_p^A
ϵ_s	error term in the expression of u_s using a truncated Taylor series expansion
$\vec{\kappa}$	the wave number vector
ϕ	the basis function used in the Finite Element method for representing u over the element
ψ	test function used in the integrals of the derivative in the Finite Element formulation
ω	the angular frequency

The following appendices 1-5 contain a listing of the program "generate-discretization" (appendix 1), including the subroutines "preparations" (appendix 2), "setup-equations" (appendix 3), "solve-equations" (appendix 4), and "analyze-solution" (appendix 5), which are an embodiment of a way to generate approximations for space derivatives. While this program is written to generate directional discretizations, minor modifications will enable it to analyze a given discretization.

The program is an input file for the program MuPAD (version 2.50), which can be found at `http://www.sciface.com`, including documentation. Note that comments in this language are between "/*" and "*/" or between pairs of #'s or on a line after //. The sequence ":=" stands for an assignment, while a command is terminated by ";" or by ":".

The embodiment is in 3 dimensions, in the Finite Difference formulation, using an

optimization of the error ϵ_n. It should be understood that the program is an example, that the invention is not limited to this specific implementation, and that extension to more dimensions, other formulations or other optimizations will pose no problem to a person skilled in numerical methods. The software can generate all the discretizations possible on a grid of given extent. The resulting discretization is in the form of a fixed stencil, and a set of stencils which can be added. The latter stencils show up in the leading term of the error of the discretization, or beyond.

The output of the run is stored in subdirectories, which are created by the program. The name of the subdirectories follows from the input parameters. The directory name is in the three-dimensional case of the form D_abc-O_d-G_ef-OP_g In this code, the lower-case letters stand for the following : a=se1, b=se2, c=se3, d=order, e=gridmin, f=gridmax, g=optimize (for an explanation of the variables, see the routines). This means that the output of every run is stored in a unique directory. E.g. the directory D_100-O_2-G_-11-OP_1 contains the output of the run with se1=1, se2=0, se3=0 (i.e. the first stream-wise derivative), order=2 (the second order approximations to the derivative) gridmin=-1, gridmax=1 (the grid extends from -1..1 in all three directions) optimize=1 (generating the basis of stencils, and some coefficients which have to be set to zero for optimizing the derivative)

The remaining appendices 6-12 contain some examples. The discretization is indicated by D_abc as explained above.

APPENDIX 1 : THE PROGRAM "GENERATE-DISCRETIZATION"

```
/*
This is the program generate-discretization.

Copyright (C) 2003  Robert Struijs.
Unauthorized use is prohibited.
This publication does not imply authorization.

Purpose :
=========

Systematic determination of discretizations of derivatives which
have directional properties, that is a reduced error for certain
preferential directions.
This program is written for three dimensions.
Extension to more dimensions or other optimizations is trivial.

Use :
=====
This is an input file for the program MuPad (version 2.50), which
can be found on http://www.sciface.com . This is a mathematical program
for symbolic analysis, with capabilities comparable to codes such as
Macsyma, Reduce, Mathematica, Axiom, ...   .
Once MuPad has been started, type :
read("generate-directional-discretization.mu")

Method :
========
Described in the patent application "A method for the numerical
simulation of a physical phenomenon using extended stencils".
Additional documentation can be found in the individual routines.
The algorithm uses the following steps :
a - write the derivative to be discretized at point (0,0,0) as a
    linear combination of the points of the stencil,
    der = sum ( coefficient*unknown ),
    which contains a discretization error epsilon,
b - the values of the points of the stencil are expressed in a
```

truncated Taylor series expansion with respect to point (0,0,0), involving space derivatives expressed in the coordinates of the structured grid,

c – a local orthogonal coordinate system (e1,e2,e3) is introduced with axis e1 along a preferential direction, and the space derivatives in the Taylor series expansion are transformed into derivatives along e1, e2 and e3,

d – all the stencils which result in a consistent approximation to the derivative are computed,

e – in the case of a directional discretization, for certain preferential directions, one of which is not along a coordinate line, minimize in the discretization error epsilon the contribution of terms due to the remaining partial derivatives. For the moment, three directions are hard-coded (but this can be changed for user input) :
- one direction along the x1 axis,
- one direction along the diagonal in the x1–x2 plane.
- one direction along the body diagonal in the x1–x2–x3 space.

Alternatively, an optimization for the error terms due to derivatives normal to the preferential direction is used.

Input :

(all the variables are integers)

1 – gridmin,gridmax : the number of nodes in one direction, i=gridmin..gridmax, gridmin < gridmax.
The case |gridmax–gridmin|>1 if gridmin*gridmax=0 is here not allowed

2 – se1,se2,se3 : the derivative to which an approximation is sought : $ds = (d/de1)^{se1} (d/de2)^{se2} (d/de3)^{se3}$.
the variables have a minimum value of 0, and se1+se2+se3 > 0

3 – order : the order of the truncation error in the approximation. minimum value : 1

4 – optimize : an optimization of the discretization.
0 = all consistent discretizations
1 = all discretizations ; for a description see the routine analyze–solutions.
2 = all diagonal discretizations ; for a description see the last part of the routine setup–equations.

5 – check : possibility to read a discretization and to check
0 = no
1 = yes, verify if the discretization can be generated with the current parameters
This parameter has for the moment no effect.

Output :

The values of the coefficients which satisfy the constraints, one of the following possibilities :
- no solution is possible (grid too small for desired accuracy) ;
- only one solution exists ;
- else an expression for the grid variables, which is stored in a directory which is created by the program, together with an expression for the grid derivatives in the form of a sum of stencils, involving degrees of freedom.

For further output, see the routine analyze–discretization.

Notation :

The point where the derivative is discretized has indices (0,0,0). The coefficients use an indexation relative to this point, and are indicated by coef(i1,i2,i3).

Example :

Find on the 3x3x3 grid with a second order error the directional discretization of ds=d^2u/del^2, that is the second local derivative. Then gridmin = –1, gridmax = 1, se1 = 2, se2 = 0, se3 = 0 and order = 2.
The approximation of the derivative ds is expressed in grid based derivatives, and the program will give for i1,i2,i3 in [–1,0,1] :
- the cxx(i1,i2,i3) i.e. the coefficients of the nodes i1,i2,i3 in the linear combination which gives the approximation of the value of d^2u/dx1^2,
- the cyy(i1,i2,i3) for d^2u/dx2^2,
- the czz(i1,i2,i3) for d^2u/dx3^2,
- the cxy(i1,i2,i3) for d^2u/(dx1*dx2).
- the cxz(i1,i2,i3) for d^2u/(dx1*dx3).
- the cyz(i1,i2,i3) for d^2u/(dx2*dx3).

```
( the notation x,y,z is used in the output for x1,x2,x3 for
  compactness )
These variables are then stored in the file soln.mu in the directory
D_200-O_2-G-11.
Furthermore, the file output.txt is created in the current directory
which contains the output which appears on the screen, including the
solution in the form of a list of stencils with degrees of freedom.

Author :
========
Robert Struijs
28 av. de Gascogne
31170 Tournefeuille
France
tel ++33 (0)5 61 06 71 39

11 October 2002 : creation
4 February 2003 : cleanup
14 August  2003 : modifications :
    eliminate the input variables verbose and tofile
    add the input variables optimize and check
    add the routine analyze-solution for printout of stencils
10 September 2003 : extend analyze-solution to generate the output
    files needed for the Authors Rights claim

=====================================================================
*/

// the 8 input parameters :

gridmin    := -1:
gridmax    :=  1:
se1        :=  1:
se2        :=  0:
se3        :=  0:
order      :=  2:
optimize   :=  0:
check      :=  0:

protocol("output.txt"):
```

```
read("preparations.mu"):
print(Unquoted,"The program has finished"):
system("xkbbell"):
# That's all, folks ! #
```

APPENDIX 2 : THE SUBROUTINE "PREPARATIONS"

```
/*
This is the subroutine preparations.

Copyright (C) 2002  Robert Struijs
Unauthorized use is prohibited.
This publication does not imply authorization.

This routine is used to check the validity of the input parameters
and to call various subroutines, which are used to determine which
stencils satisfy the constraints.

called from : generate-discretization.mu

calls : setup-equations.mu
        solve-equations.mu

input variables :
  gridmin, gridmax,
  se1, se2, se3,
  order, optimize, check

local variables :
 problems : used in checking the input parameters
 error messages
*/
TEXTWIDTH:=80:

print(Unquoted,"-----  preparations  -----"):
problems:=0:
error1a:="The value of gridmax is ".expr2text(gridmax)."\n".
         "It should be an integer.":
error1b:="The value of gridmin is ".expr2text(gridmin)."\n".
         "It should be an integer.":
error1c:="The value of gridmin is ".expr2text(gridmin)."\n".
         "It should be smaller than ".
          expr2text(gridmax)." (gridmax).":
error1d:="This is too small a grid (".expr2text(gridmin)."..".
```

```
          expr2text(gridmax)."). Think big!":
error2:="Problems with se1,se2 and se3, se1=".expr2text(se1).
        ", se2=".expr2text(se2)." and se3=".expr2text(se3)."\n".
        "They should be integers, minimum 0, one of them not zero":
error3:="The value of order is ".expr2text(order)."\n".
        "It should be an integer with a minimal value of 1":
error4:="The value of optimize is ".expr2text(optimize)."\n".
        "It should be 0, 1 or 2":
error5:="The value of check is ".expr2text(check)."\n".
        "It should be 0 or 1":

if not testtype(gridmax, Type::Integer) then
  print(Unquoted,error1a):
  problems:=1:
end_if:
if not testtype(gridmin, Type::Integer) then
  print(Unquoted,error1b):
  problems:=1:
end_if:
if gridmin >= gridmax then
  print(Unquoted,error1c):
  problems:=1:
end_if:
/* the unit cube is not allowed in this version*/
if abs(gridmax-gridmin)=1 and gridmin*gridmax=0 then
  print(Unquoted,error1d):
  problems:=1:
end_if:

print(Unquoted," gridmin = ".expr2text(gridmin).
               ", gridmax = ".expr2text(gridmax)):

if not (testtype(se1, Type::Integer) and
        testtype(se2, Type::Integer) and
        testtype(se3, Type::Integer) and se1+se2+se3 >= 1) then
  print(Unquoted,error2):
  problems:=1:
end_if:
print(Unquoted," se1 = ".expr2text(se1).
               ", se2 = ".expr2text(se2).
```

```
                    ", se3 = ".expr2text(se3)):

if not (testtype(order, Type::Integer) and order >=1) then
  print(Unquoted,error3):
  problems:=1:
end_if:
print(Unquoted," order = ".expr2text(order)):

if not (optimize=0 or optimize=1 or optimize=2) then
  print(Unquoted,error4):
  problems:=1:
end_if:
print(Unquoted," optimize = ".expr2text(optimize)):

if not (check=0 or check=1) then
  print(Unquoted,error5):
  problems:=1:
end_if:
print(Unquoted," check = ".expr2text(check)):

if problems=0 then
  read("setup-equations.mu"):
  read("solve-equations.mu"):
end_if:

# preparations are done #
```

Appendix 3 : the subroutine "setup-equations"

```
/*
This is the subroutine setup-equations.

Copyright (C) 2003  Robert Struijs
Unauthorized use is prohibited.
This publication does not imply authorization.

This routine is used to set the equations to be solved.

called from : preparations.mu
calls       : none

input :
=======
- se1,se2 and se3 : the powers in the expression
  (d/de1)^se1 (d/de2)^se2 (d/de3)^se3
- order : the order of the error in the approximation
- gridmin,gridmax : the min/max value of the grid in each
  coordinate direction

output :
========
- eqns : list of equations to be solved for the coefficients of the
  grid ; they involve all orders except the highest.
- eqndir : the list of local derivatives appearing in the error
  term, which are set to zero.
- vars : the list of variables to be solved, i.e. a list of the
  coefficients for a given node for a given grid derivative.
  examples : cz(-1, -1, -1) the weight of node (-1,-1,-1) in the
  approximation of du/dz ;  cxz(-1,0,2) is similar for
  node (-1,0,2) for the derivative d2u/dxdz.
- cp1p2p3(i1,i2,i3) : the coefficients of the nodes i1,i2,i3 of the
  grid in the linear combination which gives the approximation
  of the value of (d/dx1)^p1 (d/dx2)^p2 (d/dx3)^p3 with
  p1 = 0,1,...,hder, p2 = 0,1,...,hder-p1 and p3 = hder-p1-p2 such
  that p1 + p2 + p3 = hder (see below : method used).
  Note : the parameters p1,p2 and p3 are used in the name cp1p2p3 ;
```

the indices i1,i2 and i3 are retained as indices between brackets.

local variables : (see method used for details)

- hder, which stands for the highest derivative, se1 + se2 + se3
- het, which means highest error term, hder + order
- der(p1,p2,p3,k1,k2,k3) stands for the term d^(k1+k2+k3)/(dx1^k1 dx2^k2 dx3^k3) in the development of d^(p1+p2+p3)/(dx1^p1 dx2^p2 dx3^p3), where p1=0,1,...,hder, p2=0,1,...,hder−p1 and p3=hder−p1−p2 such that p1+p2+p3=hder. In other words, in der(p1,p2,p3,k1,k2,k3), the p1,p2,p3 indices refer to the grid derivatives we are searching, which have therefore p1+p2+p3=se1+se2+se3. The k1,k2,k3 parts come from the contribution of the grid points (i,j,k) to the derivative d^(k1+k2+k3)/(dx1^k1 dx2^k2 dx3^k3). therefore, we set der(p1,p2,p3,k1,k2,k3)=1 if k1=p1 and k2=p2 and k3=p3, else der=0.
- t11,t12,t13,t21,t22,t23,t31,t32,t33 are used in the coordinate transformation
- dsax(k1,k2,k3) is the appearance of d^(k1+k2+k3)/(dx1^k1 dx2^k2 dx3^k3) in ds
- eterm(k1,k2,k3,m1,m2,m3) : the coefficient in d^(m1+m2+m3)/(de1^m1 de2^m2) due to d^(k1+k2+k3)/(dx1^k1 dx2^k2 dx3^k3)
- dirder(m1,m2,m3) : the derivatives in ds, grouped according to d^(m1+m2+m3)/(de1^m1 de2^m2 de3^m3), the local partial derivatives.

Note :

The notation i1,i2,i3 is relative to the point 0,0,0 where the derivative is discretized.

Method used :

The input variables se1,se2 and se3 express that we are looking for a discretization which is an approximation to the local derivative ds(se1,se2,se3) = (d/de1)^se1 (d/de2)^se2 (d/de3)^se3. The local derivative ds uses the local coordinate system (e1,e2,e3) which is tied to a preferential direction. However, on the grid we have only available the grid based

derivatives dg(p1,p2,p3) = d^(p1+p2+p3)/(dx1^p1 dx2^p2 dx3^p3). A coordinate transformation expresses ds in a sum of dg's. The dg's are the derivatives which we are searching, since it are those which are implemented in a code used for a numerical solution. Note that only those dg's are used with p1 + p2 + p3 = se1 + se2 + se3 = hder. The grid based derivatives dg(p1p2,p3) are written as a linear combination of the points of the grid, where dg(p1,p2,p3)=sum1(cp1p2p3(i1,i2,i3)*u(i1,i2,i3)), sum1 is over all i1,i2,i3 of the grid, where u(i1,i2,i3) are the values of u at the nodes of the grid. Remark that we start with all the points of the grid. If the weight turns out to be zero, the resulting stencil does not use that point of the grid.

The values u(i1,i2,i3) are developed in a truncated Taylor series expansion involving grid based kth derivatives from the constant term, the first derivative ... up to the highest derivative het = hder + order.

The consistency conditions apply to all the kth derivatives up to het individually. We therefore collect the contributions to the kth derivative of the points of the grid in

der(p1,p2,p3,k1,k2,k3) = sum1{ sum2[
cp1p2p3(i1,i2,i3)*(i1^k1)*(i2^k2)*(i3^k3)/k1!/k2!/k3!] },

sum1 is over k1 + k2 + k3 = k and k=0,1,...,het,
sum2 concerns i1, i2 and i3, and is over all the points of the grid.

A grid based derivative dg(p1,p2,p3) is therefore a collection of derivatives der(p1,p2,p3,k1,k2,k3) containing coefficients cp1p2p3(i1,i2,i3).

In order to obtain a consistent discretization, constraints have to be imposed on der(p1,p2,p3,k1,k2,k3). For all terms k1,k2,k3 such that k1 + k2 + k3 < het, the contribution der(p1,p2,p3,k1,k2,k3)=0 for all k1,k2,k3 except for the case k1=p1, k2=p2 and k3=p3, when der(p1,p2,p3,k1,k2,k3)=1.

For the error term in ds, k1 + k2 + k3 = het, minimize unwanted terms for certain preferential directions after transformation to the local system (e1,e2,e3).

This requires a little bit of administration.

First, define a variable dsax(p1,p2,p3) which tells how the local derivative ds is expressed in the grid based derivatives dg(p1,p2,p3). Then, compute a variable eterm(k1,k2,k3,m1,m2,m3) which says how the grid derivatives der[p1,p2,p3,k1,k2,k3] relate to the individual local derivatives. The combination of the two

```
gives the error term of ds expressed in local derivatives
dirder(m1,m2,m3).

Therefore,
1) Compute hder = se1 + se2 + se3 and het = hder + order.
2) For all p1 + p2 + p3 = hder, we first write expressions for :
   der(p1,p2,p3,0,0,0),
   der(p1,p2,p3,1,0,0), der(p1,p2,p3,0,1,0), der(p1,p2,p3,0,0,1) ,
   and so on, up to the derivatives der(p1,p2,p3,k1,k2,k3) with
   k1 + k2 + k3 = k, and k=0,1,...,het.
   At the same time, the equations are set up, saying that all the
   der(p1,p2,p3,k1,k2,k3) = 0, except for k1=p1, k2=p2 and p3=k3
   where der(p1,p2,p3,p1,p2,p3)=1.
   These are the conditions for a consistent discretization of the
   derivatives. A list of variables is made for later use in the
   linear solver of the equations.
3) The terms der(p1,p2,p3,k1,k2,k3) with p1 + p2 + p3 = hder and
   k1 + k2 + k3 = het are the leading error terms in ds.
   We want to impose conditions in the axis system (e1,e2,e3).
   Transform the der(p1,p2,p3,k1,k2,k3) with p1 + p2 + p3 = hder
   and k1 + k2 + k3 = het to the local axis system.
   This is done in two steps :
   - translate the leading error term of ds in grid based
     derivatives
   - translate the grid based derivatives back to local
4) Set up directional equations, stating that certain dirder are
   zero.

The numbers below in the code refer to the steps just described.
*/

print(Unquoted,"-----  setup equations  -----"):
tset:=time():

/* step 1 */

hder := se1 + se2 + se3:
het  := hder + order:
print(Unquoted," Discretization of a derivative in 3D with order ".
expr2text(order)." :\n".
```

```
expr2text(se1)." times a derivation along the direction e1,\n".
expr2text(se2)." times a derivation along the direction e2,\n".
expr2text(se3)." times a derivation along the direction e3,\n".
"the highest derivative is ".expr2text(hder) ):

/*
step 2
- Loop over p1, p2 and p3 such that p1 + p2 + p3 = hder and loop
  over k1,k2 and k3 with k1 + k2 + k3 <= het.
- The factor (i1^k1)*(i2^k2)*(i3^k3)/k1!/k2!/k3! comes from the
  Taylor series.
- The equations for consistency are set up only for
  k1 + k2 + k3 < het, since the terms with k1 + k2 + k3 = het are
  treated in steps 3 and 4.
- For the linear solution procedure, a list of variables is made.
*/
print(Unquoted,
"Establishing the equations for a consistent discretization.\n"):

/*
initialize lists and arrays
The coef(...) are left undefined
*/

eqns      := []:
vars      := []:
dirnames  := ["x","y","z"]:
der       := array(0..hder,0..hder,0..hder,0..het,0..het,0..het):
strnm     := array(0..hder,0..hder,0..hder):
vararray  := array(0..hder,0..hder,0..hder,
             gridmin..gridmax,gridmin..gridmax,gridmin..gridmax):

# loop over the combinations p1 + p2 + p3 = hder #
for p3 from 0 to hder do
  for p2 from 0 to hder-p3 do
    p1:=hder-p3-p2:

// Create some convenient names for the variables .

    strnm[p1,p2,p3]:="":
```

```
    if p1 > 0 then
      for c1 from 1 to p1 do
        strnm[p1,p2,p3] := strnm[p1,p2,p3].dirnames[1]:
      end_for:
    end_if:
    if p2 > 0 then
      for c2 from 1 to p2 do
        strnm[p1,p2,p3] := strnm[p1,p2,p3].dirnames[2]:
      end_for:
    end_if:
    if p3 > 0 then
      for c3 from 1 to p3 do
        strnm[p1,p2,p3] := strnm[p1,p2,p3].dirnames[3]:
      end_for:
    end_if:
    (vararray[p1,p2,p3,i1,i2,i3] :=
      text2expr( "c".strnm[p1,p2,p3]."(".expr2text(i1).","
        .expr2text(i2).",".expr2text(i3).")"))
        $i1=gridmin..gridmax $i2=gridmin..gridmax
        $i3=gridmin..gridmax :
    vars:=vars.[vararray[p1,p2,p3,i1,i2,i3]
           $i1=gridmin..gridmax $i2=gridmin..gridmax
           $i3=gridmin..gridmax ]:

/*
Collect the contribution to the derivatives from the nodes :
this is the sum of Taylor series for points i1*Dx, i2*Dy, i3*Dz
*/

// loop over combinations k1 + k2 + k3 <= het, i.e. all orders
    for k1 from 0 to het do
      for k2 from 0 to het-k1 do
        for k3 from 0 to het-k1-k2 do
          der[p1,p2,p3,k1,k2,k3]:=
            _plus(vararray[p1,p2,p3,i1,i2,i3]*
            ((i1*Dx)^k1)*((i2*Dy)^k2)*((i3*Dz)^k3)/k1!/k2!/k3!
            $i1=gridmin..gridmax $i2=gridmin..gridmax
            $i3=gridmin..gridmax):
          if (k1=p1 and k2=p2 and k3=p3) then   // consistency
            eqns:=eqns.[subsex(der[p1,p2,p3,k1,k2,k3],
```

```
                                    Dx=1,Dy=1,Dz=1)=1]
            elif (k1+k2+k3<het) then
              eqns:=eqns.[subsex(der[p1,p2,p3,k1,k2,k3],
                                    Dx=1,Dy=1,Dz=1)=0]
            end_if:
          end_for:  // loop k3
        end_for:    // loop k2
      end_for:      // loop k1
    end_for:  // loop p2
end_for:    // loop p3

/*
step 3
We consider only k1 + k2 + k3 = het, the error terms.
The variables dirder(m1,m2,m3) represent derivatives in local
coordinates.
The grid coordinates are x1,x2 and x3, and the local coordinates
are e1, e2 and e3.
The coordinate transformation is given by
x1 = t11 e1 + t21 e2 + t31 e3
x2 = t12 e1 + t22 e2 + t32 e3
x3 = t13 e1 + t23 e2 + t33 e3
We use the two transformations :
- ds=(d/de1)^se1 (d/de2)^se2 (d/de3)^se3 is expressed in grid
  based derivatives (d/dx1)^k1 * (d/dx2)^k2 * (d/dx3)^k3
- the derivatives (d/dx1)^k1 * (d/dx2)^k2 * (d/dx3)^k3, are written
  in local coordinates.
*/

print(Unquoted,"Computing the local derivative error terms.\n"):

/* translation of ds in terms of axis derivatives */

dde1 := t11*ddx1 + t12*ddx2 + t13*ddx3:
dde2 := t21*ddx1 + t22*ddx2 + t23*ddx3:
dde3 := t31*ddx1 + t32*ddx2 + t33*ddx3:

dsax :=array(0..hder,0..hder,0..hder):
dsaxis       := poly( dde1^se1*dde2^se2*dde3^se3 , [ddx1,ddx2,ddx3]):
dsaxislist   := poly2list( dsaxis ):
```

```
dsaxisterms := nops( dsaxislist ):
for dt from 1 to dsaxisterms do
  p1:=dsaxislist[dt][2][1]:
  p2:=dsaxislist[dt][2][2]:
  p3:=dsaxislist[dt][2][3]:
  dsax[p1,p2,p3]:=dsaxislist[dt][1]:
end_for:

/* translation table for the terms der[p1,p2,p3,k1,k2,k3] from grid
   to local */

DDx1 := t11*DDe1 + t21*DDe2 + t31*DDe3:
DDx2 := t12*DDe1 + t22*DDe2 + t32*DDe3:
DDx3 := t13*DDe1 + t23*DDe2 + t33*DDe3:

eterm :=array(0..het,0..het,0..het,
              0..het,0..het,0..het):
for k1 from 0 to het do
  for k2 from 0 to het-k1 do
    k3 := het-k1-k2:
    kpoly  := poly( DDx1^k1*DDx2^k2*DDx3^k3 , [DDe1,DDe2,DDe3] ):
    klist  := poly2list( kpoly ):
    kterms := nops( klist ):
    for dk from 1 to kterms do
      m1:=klist[dk][2][1]:
      m2:=klist[dk][2][2]:
      m3:=klist[dk][2][3]:
      eterm[k1,k2,k3,m1,m2,m3]:=klist[dk][1]:
    end_for:
  end_for:
end_for:

/*
The combination of the two :
- dsax(p1,p2,p3) tells how the local derivative ds is expressed in
  the grid based derivatives dg(p1,p2,p3)
- eterm(k1,k2,k3,m1,m2,m3) says how the grid derivatives
  der[p1,p2,p3,k1,k2,k3] relate to the individual local derivatives
*/
```

```
dirder :=array(0..het,0..het,0..het):
for m1 from 0 to het do
  for m2 from 0 to het-m1 do
    m3:=het-m1-m2: // m1+m2+m3=het, only in the leading error term
    dirder[m1,m2,m3]:= 0:
    for p1 from 0 to hder do
      for p2 from 0 to hder-p1 do
        p3:=hder-p1-p2: //p1+p2+p3=hder, to grid-based derivatives
        for k1 from 0 to het do
          for k2 from 0 to het-k1 do
            k3:=het-k1-k2: // k1+k2+k3=het, from the het terms
            dirder[m1,m2,m3]:= dirder[m1,m2,m3]
            +dsax[p1,p2,p3]*der[p1,p2,p3,k1,k2,k3]/
                (Dx^p1)/(Dy^p2)/(Dz^p3)*eterm[k1,k2,k3,m1,m2,m3]:
          end_for:  // k2
        end_for:    // k1
      end_for:      // p2
    end_for:        // p1
  end_for:          // m2
end_for:            // m1

/*
Decide on which directional derivatives are to be set to zero.
*/
eqndir :=[]:

if   (se1=hder) then
  # pure derivative along e1 #
  print(Unquoted,
    "Equations for optimizing a pure derivative along e1"):
  for m1 from 0 to het do
    for m2 from 0 to het-m1 do
      m3:=het-m1-m2:
      if not ( m1=(se1+order) ) then
        eqndir:=eqndir.[dirder[m1,m2,m3]=0]
      end_if:
    end_for:
  end_for:
elif (se2=hder) then
  # pure derivative along e2 #
```

```
print(Unquoted,
  "Equations for optimizing a pure derivative along e2"):
for m1 from 0 to het do
  for m2 from 0 to het-m1 do
    m3:=het-m1-m2:
    if not ( m2=(se2+order) ) then
      eqndir:=eqndir.[dirder[m1,m2,m3]=0]
    end_if:
  end_for:
end_for:
elif (se3=hder) then
  # pure derivative along e3 #
  print(Unquoted,
    "Equations for optimizing a pure derivative along e3"):
  for m1 from 0 to het do
    for m2 from 0 to het-m1 do
      m3:=het-m1-m2:
      if not ( m3=(se3+order) ) then
        eqndir:=eqndir.[dirder[m1,m2,m3]=0]
      end_if:
    end_for:
  end_for:
elif (se1=0) then
  # mixed derivative involving e2,e3#
  print(Unquoted,
   "Equations for optimizing a mixed derivative involving e2,e3"):
  for m1 from 0 to het do
    for m2 from 0 to het-m1 do
      m3:=het-m1-m2:
      if not (   ( m2=(se2+order) and m3=se3 ) or
                 ( m2=se2 and m3=(se3+order) )
             ) then
        eqndir:=eqndir.[dirder[m1,m2,m3]=0]
      end_if:
    end_for:
  end_for:
elif (se2=0) then
  # mixed derivative involving e1,e3#
  print(Unquoted,
   "Equations for optimizing a mixed derivative involving e1,e3"):
```

```
for m1 from 0 to het do
  for m2 from 0 to het-m1 do
    m3:=het-m1-m2:
    if not (   ( m1=(se1+order) and m3=se3  ) or
               ( m1=se1  and m3=(se3+order) )
           ) then
      eqndir:=eqndir.[dirder[m1,m2,m3]=0]
    end_if:
  end_for:
end_for:
elif (se3=0) then
  # mixed derivative involving e1,e2#
  print(Unquoted,
   "Equations for optimizing a mixed derivative involving e1,e2"):
  for m1 from 0 to het do
    for m2 from 0 to het-m1 do
      m3:=het-m1-m2:
      if not (   ( m2=(se2+order) and m1=se1  ) or
                 ( m2=se2  and m1=(se1+order) )
             ) then
        eqndir:=eqndir.[dirder[m1,m2,m3]=0]
      end_if:
    end_for:
  end_for:
else
  # mixed derivative involving all axes#
  print(Unquoted,
   "Equations for optimizing a mixed derivative involving e1,e2,e3"):
  for m1 from 0 to het do
    for m2 from 0 to het-m1 do
      m3:=het-m1-m2:
      if not (   ( m1=(se1+order) and m2=se2 and m3=se3  ) or
                 ( m1=se1  and m2=(se2+order) and m3=se3  ) or
                 ( m1=se1  and m2=se2  and m3=(se3+order) )
             ) then
        eqndir:=eqndir.[dirder[m1,m2,m3]=0]
      end_if:
    end_for:
  end_for:
end_if:
```

```
/*
find the derivatives which are of no interest
*/
eqnuninteresting := []:
for p3 from 0 to hder do
  for p2 from 0 to hder-p3 do
    p1:=hder-p2-p3:
    for i1 from gridmin to gridmax do
      for i2 from gridmin to gridmax do
        for i3 from gridmin to gridmax do
          if   p1<>0 and p2=0 and p3=0 then
            if i2<>0 or i3<>0 then
              eqnuninteresting := eqnuninteresting.
              [vararray[p1,p2,p3,i1,i2,i3]=0]:
            end_if:
          elif p2<>0 and p3=0 and p1=0 then
            if i3<>0 or i1<>0 then
              eqnuninteresting := eqnuninteresting.
              [vararray[p1,p2,p3,i1,i2,i3]=0]:
            end_if:
          elif p3<>0 and p1=0 and p2=0 then
            if i1<>0 or i2<>0 then
              eqnuninteresting := eqnuninteresting.
              [vararray[p1,p2,p3,i1,i2,i3]=0]:
            end_if:
          end_if:
        end_for:
      end_for:
    end_for:
  end_for:
end_for:

print(Unquoted,"The equations are ready to be solved."):
print(Unquoted,"the time spent is "
  .expr2text(float((time()-tset)/1000))." s."):

# setup-equations is done #
```

Appendix 4 : the subroutine "solve-equations"

```
/*
This is the subroutine solve-equations.

Copyright (C) 2003  Robert Struijs
Unauthorized use is prohibited.
This publication does not imply authorization.

In this routine, we search the solution of the adirectional
and the directional equations.
We choose here the axis x1 and the diagonals x1-x2 and x1-x2-x3.
Other choices are possible, e.g along x1=2 x2 etc.

called from : preparations.mu
calls       : analyze.mu

input :
=======
- eqns : the adirectional equations for the discretization of a
  derivative
- eqndir : the directional equations directions, to be set to zero
  for certain directions.
- se1,se2 and se3 : the powers in the expression
  (d/de1)^se1 (d/de2)^se2 (d/de3)^se3
- order : the order of the error in the approximation
- optimize : the type of optimization
- gridmin,gridmax : the min/max value of the grid in each
  coordinate direction

*/

print(Unquoted,"-----  solve equations  -----"):
tsol:=time():
dir := "D_".expr2text(se1).expr2text(se2).expr2text(se3).
       "-O_".expr2text(order).
       "-G_".expr2text(gridmin).expr2text(gridmax).
       "-OP_".expr2text(optimize):
system("mkdir ".dir):
```

```
impossible :=0:
imp:="No solution was found":

prefdirnames:=[
"of consistent discretizations",
"with only streamwise derivatives for e1 // x1-axis",
"with only streamwise derivatives for e1 // diagonal x1-x2",
"with only streamwise derivatives for e1 // diagonal x1-x2-x3"]:

/*
The coordinate transformation matrix for expressions in the
local basis B
*/

assume(a,Type::Real):
assume(b,Type::Real):
assume(c,Type::Real):
gentrans     := [ t11 =  cos(b)*cos(a),
                  t12 =  cos(b)*sin(a),
                  t13 =  sin(b),
                  t21 = -cos(c)*sin(a)+cos(a)*sin(c)*sin(b),
                  t22 =  cos(c)*cos(a)+sin(a)*sin(c)*sin(b),
                  t23 = -sin(c)*cos(b),
                  t31 = -sin(a)*sin(c)-cos(c)*cos(a)*sin(b),
                  t32 =  cos(a)*sin(c)-cos(c)*sin(a)*sin(b),
                  t33 =  cos(c)*cos(b)]:

dirx1      := subsex(gentrans,[c=0,b=0,a=0]):
dirx2      := subsex(gentrans,[c=0,b=0,a=PI/4]):
dirx3      := subsex(gentrans,[c=0,b=arctan(sin(PI/4)),a=PI/4]):
/*
This is for any angle gamma (c), i.e. any orientation of
the e2-e3 axes, and only the e1-axis aligned.
No difference has been found for any of the stencils, so far ...
And the above is quite a bit faster.
dirx1      := subsex(gentrans,[b=0,a=0]):
dirx2      := subsex(gentrans,[b=0,a=PI/4]):
dirx3      := subsex(gentrans,[b=arctan(sin(PI/4)),a=PI/4]):
*/
```

```
extra:=[]:
/*
It is possibl to add some extra conditions by reading
the file extra.mu. Uncomment if needed.
*/
//read("extra.mu"):

/*
setting the error to 0 for terms < order : consistency
*/

print(Unquoted,
  "Solving the equations for consistent approximations"):
solnadir  := linsolve(eqns.extra,vars):
if solnadir=FAIL then
  impossible :=1:
  print(Unquoted,imp):
end_if:

if optimize=2 then

/*
setting the error to 0 for direction 1
*/

if impossible=0 then
  print(Unquoted,"Solving the directional equations - along x1"):
  eqndir1  := subsex(eqndir,dirx1,Dx=1,Dy=1,Dz=1):
  solndir1 := linsolve(solnadir.eqndir1,vars):
  if solndir1=FAIL then
    impossible :=1:
    print(Unquoted,imp):
  end_if:
end_if:

/*
setting the error to 0 for direction 2
*/

```

```
if impossible=0 then
  print(Unquoted,
       "Solving the directional equations - along diag x1-x2"):
  eqndir2  := map(subsex(eqndir,dirx2,Dx=1,Dy=1,Dz=1),factor):
  solndir2 := linsolve(solndir1.eqndir2,vars):
  if solndir2=FAIL then
    impossible :=1:
    print(Unquoted,imp):
  end_if:
end_if:

/*
setting the error to 0 for direction 3
warning : do not simplify the transformation matrix,
or MuPad may give a false FAIL
 */

if impossible=0 then
  print(Unquoted,
    "Solving the directional equations - along diag x1-x2-x3\n"):
  eqndir3  := map(subsex(eqndir,dirx3,Dx=1,Dy=1,Dz=1),factor):
  solndir3 := linsolve(solndir2.eqndir3,vars):
  if solndir3=FAIL then
    impossible :=1:
    print(Unquoted,imp):
  end_if:
end_if:

end_if: // optimize=2

/*
Write the solution of the equations to a file if desired
*/
if impossible=0 then
  if optimize=2 then
    write(Text,dir."/soln.mu",solnadir,solndir1,solndir2,solndir3):
  else
    write(Text,dir."/soln.mu",solnadir):
  end_if:
  print(Unquoted,
```

```
      "The solution is written to the file : ".dir."/soln.mu"):
else
  system("rmdir ".dir ):
end_if:

/*
Analyze the result
*/
print(Unquoted,"the time spent is "
.expr2text(float((time()-tsol)/1000))." s."):

if ( impossible=0 ) then
  read("analyze-solution.mu"):
end_if:

# solve-equations is done #
```

Appendix 5 : The subroutine “analyze-solution”

```
/*
This is the subroutine analyze-solution

Copyright (C) 2003  Robert Struijs
Unauthorized use is prohibited.
This publication does not imply authorization.

In this routine, we analyze the solution to the constraints
as defined in the input file generate-discretizations.mu.

called from : solve-equations.mu
calls       : none

input :
=======
- solnadir,solndir1,solndir2,solndir3 :
  the various solutions obtained in the routine solve-equations
- se1,se2 and se3 : the powers in the expression
  (d/de1)^se1 (d/de2)^se2 (d/de3)^se3
- order : the order of the error in the approximation
- gridmin,gridmax : the min/max value of the grid in each
  coordinate direction

output :
========
The grid-based approximations in the form of stencils with free
parameters, for the consistent approximations, optimize=0 or
optimize=1 ; additionally, for optimize=1, the conditions are
listed which eliminate the contribution in the error normal to
the preferential direction.
For optimize=2, only the stencils with degrees of freedom are given
which eliminate certain error terms for the case that the
preferential direction is along the x-axis or along diagonals, see
the routine setup-equations.
These results are also stored in the file output.txt.

In a directory created by the program in the routine
```

• solve-equations, some additional MuPad files are created in the case that optimize=1.

variables :

These are mainly the variables as already defined in the calling routines.

Local variables :

- gentrans : the coefficients of the transformation between the grid and the local basis B ; the terms tij are described in the patent
- dernames : a list of names of derivatives which are
  du/de1 -> dus
  du/de2 -> dun
  du/de3 -> dut
  and combinations thereof for higher derivatives, e.g. d2udsdt
- indexpart[p1,p2,p3] : the grid indices for the variables
- thesol : the solution which is analyzed, one of the solutions generated by solve-equations
- fixedstencil : the part of the approximation which does not involve a parameter
- varstencil : the part of the approximation which does involves parameters
- stencil[p1,p2,p3,n] : the nth stencil associated to a parameter
- vartok : rewrite the variables which are used in setup and solv, the coef(i,j,k), in variables k... associated to the stencils expressing the degrees of freedom.
- ktovar : the inverse of the above
- varpart[p1,p2,p3] : those variables related to the approximation of the p1,p2,p3 grid derivative
- gensol : the general solution expressed in stencil and in fixedstencil, describing the solutions allowed by the input parameters
- soltotu : the part of the solution which is uninteresting
- dirder contains the error in the approximation split in the components of order het with respect to the local basis B
  The solution, the transformation and the k-variables can

```
        be substituted into dirder before use.

    Note :
    ======
    The notation i1,i2,i3 is relative to the point 0,0,0 where the
    derivative is discretized.

    Actions :
    =========

    1) ask which output has to be generated
    2) find the stencils which appear in the solution
    3) if optimize=1 and a pure derivative, choose which error terms
       in the solution should be zero
    4) prepare some output files with the results
       This takes place throughout this routine.
    The numbers below in the code refer to the steps just described.
    */

    print(Unquoted,"-----  analyze solution  -----"):
    tanalyze:=time():

    answerbasis    := dunno:
    answerbasison  := dunno:
    answererror    := dunno:

    /* you can preset manually (by editing this file) the answers
    answerbasis    := y:
    answerbasison  := y:
    answererror    := y:
    */

    /* the questions */

    while not (answerbasis=y or answerbasis=n) do:
      input("Write separately the stencils (basis) ? (y/n)",
            answerbasis):
    end_while:

    while not (answerbasison=y or answerbasison=n) do:
```

```
    input("Compute the orthogonal stencils (basis) ? (y/n)",
          answerbasison):
  end_while:

  while not (answererror=y or answererror=n) do:
    input("Compute the error terms ? (y/n)",answererror):
  end_while:

  if answererror=y then
    answerfortran:=dunno:
  else
    answerfortran:=n:
  end_if:

  /* you can preset manually the answer
  answerfortran:=n:
  */

  while not (answerfortran=y or answerfortran=n) do:
    input("Fortran output of the errors ? (y/n)",answerfortran):
  end_while:


  //==========================================================

  if optimize=2 then
    thesol := solndir3:
    txtsol:="optimized":
  elif optimize=0 or optimize=1 then
    thesol := solnadir:
    txtsol:="consistent":
  end_if:


  if answerbasis=y then

  /* step 2
  find the stencils which appear in the solution :
  1. substitute in the vars the solution.
  2. loop over the vars.
```

```
*/

/* procedure for creating an equation, v1=v2 */
eq := proc(v1,v2)
begin
v1=v2:
end_proc:

cnstflag:=1:
solu:=linsolve(thesol.eqnuninteresting,vars):
if solu=FAIL then
  cnstflag:=0:
end_if:

uninsol:=[]:
vars0   := map(vars,eq,0):
nv      := nops(vars):
ivpflag:=array(1..nv,(i)=0$i=1..nv):

for p3 from 0 to hder do
  for p2 from 0 to hder-p3 do
    p1  :=hder-p3-p2:   // loop over the grid-based derivatives

    nst[p1,p2,p3] := 0:
    varlist[p1,p2,p3]:=[]:
    varpart[p1,p2,p3] := [vararray[p1,p2,p3,i1,i2,i3]
                          $i1=gridmin..gridmax
                          $i2=gridmin..gridmax
                          $i3=gridmin..gridmax ]:
    nvp := nops(varpart[p1,p2,p3]):
    indexpart[p1,p2,p3] := [op(varpart[p1,p2,p3][i])] $i=1..nvp:
    solpart[p1,p2,p3]       := subs(varpart[p1,p2,p3],thesol):
    fixedstencil[p1,p2,p3] := subs(solpart[p1,p2,p3],vars0):
    varstencil[p1,p2,p3]    := zip(solpart[p1,p2,p3],
                                   fixedstencil[p1,p2,p3],_subtract):
    if cnstflag=1 then
      solpartu[p1,p2,p3]       := subs(varpart[p1,p2,p3],solu):
      fixedstencilu[p1,p2,p3] := subs(solpartu[p1,p2,p3],vars0):
      varstencilu[p1,p2,p3]    := zip(solpartu[p1,p2,p3],
                                      fixedstencilu[p1,p2,p3],_subtract):
```

```
    end_if:
    for iv from 1 to nv do
      steq := []:
      for iw from 1 to nv do
        if (iv = iw ) then
          steq := steq.[eq(vars[iw],1)]:
        else
          steq := steq.[eq(vars[iw],0)]:
        end_if:
      end_for:
      flag:=0:
      sttmp := (subs(varstencil[p1,p2,p3],steq)):
      for j from 1 to nvp do
        if (sttmp[j]<>0) then
          flag:=1:
        end_if:
      end_for:
      if flag=1 then
        nst[p1,p2,p3] := nst[p1,p2,p3] + 1 :
        stencil[p1,p2,p3,nst[p1,p2,p3]] := sttmp:
        ivp[p1,p2,p3,nst[p1,p2,p3]]:=iv:
        ivpflag[iv]:=1:
      end_if:
    end_for: // loop iv
  end_for:   // loop p2
end_for:     // loop p3

// contruct the k vars in clusters of one p1,p2,p3 derivative
ncl := nv/nvp: // the number of p1,p2,p3 clusters
nctr:=0:
kvarsall:=[]:
vartokall:=[]:
ktovarall:=[]:
for p3 from 0 to hder do
  for p2 from 0 to hder-p3 do
    p1  :=hder-p3-p2:   // loop over the grid-based derivatives
    nstart:=nctr*nvp+1:
    nstcum[p1,p2,p3] := 0:
    kvars[p1,p2,p3]   := []:
    ktovar[p1,p2,p3] := []:
```

```
    vartok[p1,p2,p3] := []:
    for i from nstart to nstart+nvp-1 do
      if ivpflag[i]=1 then
        nstcum[p1,p2,p3] := nstcum[p1,p2,p3]+1:
        kvars[p1,p2,p3]  := kvars[p1,p2,p3].
     [text2expr("k".strnm[p1,p2,p3].expr2text(nstcum[p1,p2,p3]))]:
        ktovar[p1,p2,p3] := ktovar[p1,p2,p3].
          [op(kvars[p1,p2,p3],nstcum[p1,p2,p3])=vars[i]]:
        vartok[p1,p2,p3] := vartok[p1,p2,p3].
            [vars[i]=op(kvars[p1,p2,p3],nstcum[p1,p2,p3])]:
      end_if:
    end_for:
    nctr:=nctr+1:
  kvarsall   :=kvarsall.kvars[p1,p2,p3]:
  ktovarall :=ktovarall.ktovar[p1,p2,p3]:
  vartokall :=vartokall.vartok[p1,p2,p3]:
  end_for:   // loop p2
end_for:     // loop p3

/*
create the basis of stencils
express the uninteresting solution in the basis of stencils
*/
for p3 from 0 to hder do
  for p2 from 0 to hder-p3 do
    p1  :=hder-p3-p2:   // loop over the grid-based derivatives
    kt:=map(stencil[p1,p2,p3,i],_mult,
      subsex(vars[ivp[p1,p2,p3,i]],vartokall)) $i=1..nst[p1,p2,p3]:
    if nst[p1,p2,p3]>1 then
      gensol[p1,p2,p3]:= [_plus(fixedstencil[p1,p2,p3][i],kt[j][i]
                           $j=1..nst[p1,p2,p3]) $i=1..nvp]:
      basis[p1,p2,p3]  := [_plus(kt[j][i]
                           $j=1..nst[p1,p2,p3]) $i=1..nvp]:
    else
      gensol[p1,p2,p3]:= zip(fixedstencil[p1,p2,p3],kt,_plus):
      basis[p1,p2,p3]  := [kt[i] $i=1..nvp]:
    end_if:
    if cnstflag=1 then
      eqtotu[p1,p2,p3]:= [gensol[p1,p2,p3][i]=solpartu[p1,p2,p3][i]
                          $i=1..nvp]:
```

```
        if hder=1 then
          tmpsol:= linsolve(eqtotu[p1,p2,p3]):
          if tmpsol=FAIL then
            cnstflag:=0:
            cnstr[p1,p2,p3]:=0:
          else
            uninsol:=uninsol.tmpsol:
            cnstr[p1,p2,p3]:=1:
            soltotu[p1,p2,p3]:=
              expr2text(op(subsex(tmpsol,vartok[p1,p2,p3]))):
          end_if:
        end_if
      end_if:
    end_for:    // loop p2
  end_for:      // loop p3
  uninsolk:=subs(uninsol,vartokall):

  /* procedure for printing a stencil */
  prst := proc(nodes,stencil)
  begin
    toprint:="":
    nps:=nops(stencil):
    prev:=0:
    for np from 1 to nps do
      wi := op(stencil,[np]):
      if wi<>0 then
        if prev=1 then
          toprint:=toprint.", ":
        end_if:
        prev:=1:
        n1:=expr2text(op(nodes,[np,1])):
        n2:=expr2text(op(nodes,[np,2])):
        n3:=expr2text(op(nodes,[np,3])):
        toprint:=toprint."{(".n1.",".n2.",".n3."),".expr2text(wi)."}":
      end_if:
    end_for:
  end_proc:

  txt1:=
   "The constraints specified in the file generate-discretization,\n":
```

```
txt2:=
"are satisfied by the following solution, which is written\n".
"using the notation k... for a free coefficient\n".
"and where T... represents a stencil which is given by a list\n".
"of nodes with corresponding weights in the notation {(i,j,k),w}\n".
"where the node is given by the indices i,j,k and the weight\n".
"by w, and where Dx, Dy and Dz are the grid spacings\n".
"in the three coordinate directions,\n".
"in the following approximations :\n".
"( a line which ends with \\ is continued on the next line)":
print(Unquoted,txt1):
print(Unquoted," gridmin = ".expr2text(gridmin).
               ", gridmax = ".expr2text(gridmax).",",):
print(Unquoted," se1 = ".expr2text(se1).
               ", se2 = ".expr2text(se2).
               ", se3 = ".expr2text(se3).","):
print(Unquoted," order = ".expr2text(order).","):
print(Unquoted," optimize = ".expr2text(optimize).","):
print(Unquoted," check = ".expr2text(check).","):
print(Unquoted,txt2):

for p3 from 0 to hder do
  for p2 from 0 to hder-p3 do
    p1  :=hder-p3-p2:
    if hder=1 then
      dertmp := "D".strnm[p1,p2,p3]." du/d".strnm[p1,p2,p3]:
    else
      dertmp :=
        stringlib::subs(stringlib::subs(stringlib::subs(
           strnm[p1,p2,p3]," x"="Dx "),"y"="Dy "),"z"="Dz ").
        "d".expr2text(hder)."u/".
        stringlib::subs(stringlib::subs(stringlib::subs(
           strnm[p1,p2,p3]," x"="dx")," y"="dy")," z"="dz"):
    end_if:
    print(Unquoted,dertmp." = Tf".strnm[p1,p2,p3]." +"):
    if nst[p1,p2,p3]>1 then
      print(Unquoted,hold(_plus)(text2expr(
        expr2text(subsex(vars[ivp[p1,p2,p3,i]],vartokall)).
           "*T".strnm[p1,p2,p3].expr2text(i))
                $i=1..nst[p1,p2,p3])," where"):
```

```
      else
        print(Unquoted, text2expr(
          expr2text(subsex(vars[ivp[p1,p2,p3,1]], vartokall)).
            "*T".strnm[p1,p2,p3].expr2text(1))
                ," where"):
      end_if:
      prst([indexpart[p1,p2,p3]], fixedstencil[p1,p2,p3]):
      print(Unquoted,"Tf".strnm[p1,p2,p3]." = ".toprint.";"):
      for i from 1 to nst[p1,p2,p3] do
        prst([indexpart[p1,p2,p3]], stencil[p1,p2,p3,i]):
        print(Unquoted,"T".strnm[p1,p2,p3].expr2text(i).
          " = ".toprint.";"):
      end_for:
    end_for:     // loop p2
  end_for:       // loop p3

  if cnstflag=1 and hder=1 then
    print(Unquoted,
    "The grid-aligned one-dimensional solution which is excluded\n".
    "from the patent is given by "):
    flag:=0:
    for p3 from 0 to hder do
      for p2 from 0 to hder-p3 do
        p1  :=hder-p3-p2:
        if cnstr[p1,p2,p3]=1 then
          if flag=1 then
            print(Unquoted," and "):
          end_if:
          flag:=1:
          print(Unquoted,soltotu[p1,p2,p3]):
        end_if:
      end_for:    // loop p2
    end_for:      // loop p3
  end_if:


//===========================================================================

/* step 3 : choose which error terms in the solution
            should be zero */
```

```
if optimize=1 and (se1=hder or se2=hder or se3=hder) then

  txt3:=
  "We use the notation\n".
  "a for the angle alpha, the rotation around the grid z-axis,\n".
  "b for the angle beta,  the rotation around the new y'-axis,\n".
  "c for the angle gamma, the rotation around newest x''-axis,\n".
  "which is the axis e1 of the local basis B, as explained in the\n".
  "patent application.\n".
  "The present optimization imposes constraints on the error terms.\n".
  "The error terms in the local basis are written to the files\n".
  "direrror.mu and direrror.txt.\n".
  "The notation Epqr means the error in the approximation related to\n".
  "the mixed pth derivative w.r.t. direction e1,\n".
  "the qth derivative w.r.t. to e2 and the rth derivative w.r.t. e3\n".
  "e.g. E201 is the contribution related to the mixed 2nd derivative\n".
  "along axis e1, and the first derivative along axis e3\n":

  print(Unquoted,txt3):

  for k1 from 0 to het do
    for k2 from 0 to het-k1 do
      k3:=het-k1-k2:
      if (se1=hder and k1=0) or (se2=hder and k2=0) or
         (se3=hder and k3=0) then
        print(Unquoted,"The error term E".
               expr2text(k1).expr2text(k2).expr2text(k3)." = 0"):
      end_if:
    end_for:    // loop k2
  end_for:      // loop k1

end_if:  // optimize=1 for pure derivative

end_if: // answerbasis ; first use

protocol():
system("mv output.txt ".dir):
print(Unquoted," The remainder of the output which appears on the\n".
"screen concerns the output which is written to the various .txt files"):
```

```
//===================================================================

/* step 4
Write the results to some files. For details, see the Readme.txt file.
*/

// a) has just ended ; b) is already done in solve-equations

// c) The solution in terms c...(i,j,k)
protocol(dir."/soln.txt"):
print(Unquoted,"The ".txtsol." solution is given by "):
map(thesol,print):
protocol():

if answerbasis=y then

//d) and e) The solution expressed in the coefficients k of the stencils
thesolk:=vartokall.subsex(thesol,vartokall):
protocol(dir."/solk.txt"):
print(Unquoted,"The ".txtsol." solution expressed in the k's is"
  ."given by "):
map(thesolk,print):
write(Text,dir."/solk.mu",thesolk):
protocol():

// f) and g) The stencils. (basis)

nb := fopen(dir."/basis.mu",Append,Text):
write(nb,indexpart,fixedstencil,stencil):
fclose(nb):

txt4:=     " with the following list of nodes and stencils:\n".
     "(the first weight of the stencil corresponds to the first node,\n".
     "the second weight with the second node, and so on)":
print(Unquoted,"The stencils in slightly different notation".
       " (including 0's)"):
protocol(dir."/basis.txt"):
for p3 from 0 to hder do
```

```
      for p2 from 0 to hder-p3 do
        p1  :=hder-p3-p2:
        if hder=1 then
          dertmp := "D".strnm[p1,p2,p3]." du/d".strnm[p1,p2,p3]:
        else
          dertmp :=
            stringlib::subs(stringlib::subs(stringlib::subs(
              strnm[p1,p2,p3],"x"="Dx "),"y"="Dy "),"z"="Dz ").
            "d".expr2text(hder)."u/".
            stringlib::subs(stringlib::subs(stringlib::subs(
              strnm[p1,p2,p3],"x"="dx"),"y"="dy"),"z"="dz"):
        end_if:
        print(Unquoted,dertmp." = Tf".strnm[p1,p2,p3]." +"):
        if nst[p1,p2,p3]>1 then
          print(Unquoted,hold(_plus)(text2expr(
            expr2text(subsex(vars[ivp[p1,p2,p3,i]], vartokall)).
              "*T".strnm[p1,p2,p3].expr2text(i))
                  $i=1..nst[p1,p2,p3]),"\n".txt4):
        else
          print(Unquoted,text2expr(
            expr2text(subsex(vars[ivp[p1,p2,p3,1]], vartokall)).
              "*T".strnm[p1,p2,p3].expr2text(1)),"\n".txt4):
        end_if:
        print(Unquoted," The nodes of the grid are ",indexpart[p1,p2,p3]):
        print(Unquoted,"Tf".strnm[p1,p2,p3]." = "):
        print(fixedstencil[p1,p2,p3]):
        for i from 1 to nst[p1,p2,p3] do
          print(Unquoted,"T".strnm[p1,p2,p3].expr2text(i).
            " = "):
          print(stencil[p1,p2,p3,i]):
        end_for:
      end_for:    // loop p2
    end_for:      // loop p3
    protocol():

    end_if : // answerbasis


    // h) and i) The orthonormal stencils ;

```

```
if answerbasison=y then

/*
create a orthonormal basis with Gramm-Schmidt and the dot product
*/

/* procedure for the generalized dot product */
dpg := proc(s1,s2)
begin
_plus(s1[i]*s2[i] $i=1..nops(s1)):
end_proc:

for p3 from 0 to hder do
  for p2 from 0 to hder-p3 do
    p1  :=hder-p3-p2:
    for ns1 from 1 to nstcum[p1,p2,p3] do
      oncomp:= stencil[p1,p2,p3,ns1]:
      for ns2 from 1 to ns1-1 step 1 do
        dp:=dpg(onstencil[p1,p2,p3,ns2],stencil[p1,p2,p3,ns1]):
        sttmp:=map(onstencil[p1,p2,p3,ns2],_mult,dp):
        oncomp:=zip(oncomp,sttmp,_subtract):
      end_for: // loop ns2
      ldp[p1,p2,p3,ns1]:=dpg(oncomp,oncomp):
      onstencil[p1,p2,p3,ns1]:=map(oncomp,_divide,
                                  sqrt(ldp[p1,p2,p3,ns1])):
    end_for: // loop ns1
  end_for:   // loop p2
end_for:     // loop p3

// just a check on some basis stencils
p1:=hder:p2:=0:p3:=0:
nstot:=nst[p1,p2,p3]:
dpar:=array(1..nstot,1..nstot):
dparon:=array(1..nstot,1..nstot):
for ns1 from 1 to nstot do
  for ns2 from 1 to nstot do
  dpar[ns1,ns2]:=dpg(stencil[p1,p2,p3,ns1],stencil[p1,p2,p3,ns2]):
  dparon[ns1,ns2]:=dpg(onstencil[p1,p2,p3,ns1],onstencil[p1,p2,p3,ns2]):
  end_for:
end_for:
```

```
nb := fopen(dir."/basison.mu",Append,Text):
write(nb,indexpart,fixedstencil,onstencil):
fclose(nb):

protocol(dir."/basison.txt"):

print(Unquoted,"The orthogonal stencils in slightly different"
." notation\n".
" (including 0's), without the stencils corresponding to the".
" coefficients\n".
"associated with other grid-based derivatives"):
for p3 from 0 to hder do
  for p2 from 0 to hder-p3 do
    p1  :=hder-p3-p2:
    if hder=1 then
      dertmp := "D".strnm[p1,p2,p3]." du/d".strnm[p1,p2,p3]:
    else
      dertmp :=
        stringlib::subs(stringlib::subs(stringlib::subs(
           strnm[p1,p2,p3],"x"="Dx "),"y"="Dy "),"z"="Dz ").
        "d".expr2text(hder)."u/".
        stringlib::subs(stringlib::subs(stringlib::subs(
           strnm[p1,p2,p3],"x"="dx"),"y"="dy"),"z"="dz"):
    end_if:
    print(Unquoted,dertmp." = Tf".strnm[p1,p2,p3]." +"):
    if nst[p1,p2,p3]>1 then
      print(Unquoted,hold(_plus)(text2expr(
        expr2text(subsex(vars[ivp[p1,p2,p3,i]],vartokall)).
           "*T".strnm[p1,p2,p3].expr2text(i))
                $i=1..nst[p1,p2,p3]),"\n".txt4):
    else
      print(Unquoted,text2expr(
        expr2text(subsex(vars[ivp[p1,p2,p3,1]],vartokall)).
           "*T".strnm[p1,p2,p3].expr2text(1)),"\n".txt4):
    end_if:
    print(Unquoted," The nodes of the grid are ",indexpart[p1,p2,p3]):
    print(Unquoted," Tf".strnm[p1,p2,p3]." = "):
    print(fixedstencil[p1,p2,p3]):
    for i from 1 to nstcum[p1,p2,p3] do
```

```
        print(Unquoted,"T".strnm[p1,p2,p3].expr2text(i).
          " = "):
        print(onstencil[p1,p2,p3,i]):
      end_for:
    end_for:   // loop p2
  end_for:     // loop p3
  protocol():

  end_if: //answerbasison

  // j) and k) The directional error terms ;

  if answererror=y then

  protocol(dir."/direrror.txt"):
  print(Unquoted,
    "The error terms in the local basis are given below\n".
    "The notation Epqr means the error in the approximation related to\n".
    "the mixed pth derivative w.r.t. direction e1,\n".
    "the qth derivative w.r.t. to e2 and the rth derivative w.r.t. e3\n".
    "e.g. E201 is the contribution related to the mixed 2nd derivative\n".
    "along axis e1, and the first derivative along axis e3\n"):

  /*
  use dirder together with the solution
  */
  direrror :=array(0..het,0..het,0..het):
  for k1 from 0 to het do:
    for k2 from 0 to het-k1 do
      k3:=het-k1-k2:
      print(Unquoted,"The error term E".
            expr2text(k1).expr2text(k2).expr2text(k3)):
      PRETTYPRINT:=FALSE:
      direrrtmp:=
        expand(subsex(dirder[k1,k2,k3],gentrans,thesol,vartokall)):
      dep:=poly(direrrtmp,[cos(a),sin(a),cos(b),sin(b),cos(c),sin(c)]):
      np0:=nops(op(dep,1)):
      for n from 1 to np0-1 do
        print(Unquoted,expr2text(op(dep,[1,n]))."+\n"):
      end_for:
```

```
      print(Unquoted,expr2text(op(dep,[1,np0]))."\n"):
      PRETTYPRINT:=TRUE:
      if answerfortran=y then
        direrror[k1,k2,k3] := direrrtmp:
      end_if:
      nb := fopen(dir."/direrror".
            expr2text(k1).expr2text(k2).expr2text(k3).".mu",Append,Text):
      write(nb,direrrtmp):
      fclose(nb):
    end_for:  // loop k2
  end_for:    // loop k1
  protocol():

  end_if: //answererror

  if answerfortran=y then

  /* this part of analyze can be activated manually */

  print(Unquoted,"The errors in FORTRAN form"):
  TEXTWIDTH:=73:
  PRETTYPRINT:=FALSE:
  MAXDEPTH:=100000:
  if optimize=0 or optimize=1 then
    kname:="kunint-opt01":
  elif optimize=2 then
    kname:="kunint-opt2":
  end_if:
  protocol(dir."/".kname.".f"):
  if nops(uninsolk)>0 then
    print(Unquoted,generate::fortran(uninsolk)):
  end_if:
  protocol():
  for k1 from 0 to het do:
    for k2 from 0 to het-k1 do
      k3:=het-k1-k2:
      tmpfort:=subsex(direrror[k1,k2,k3],[Dx=1,Dy=1,Dz=1]):
      if optimize=0 or optimize=1 then
        cname:="c".expr2text(k1).expr2text(k2).expr2text(k3)."opt01":
      elif optimize=2 then
```

```
        cname:="c".expr2text(k1).expr2text(k2).expr2text(k3)."opt2":
      end_if:
      protocol(dir."/".cname.".f"):
      print(Unquoted,generate::fortran(text2expr(cname)=tmpfort)):
      protocol():
    end_for:  // loop k2
  end_for:    // loop k1
  PRETTYPRINT:=TRUE:
  TEXTWIDTH:=80:

  //end of the FORTRAN output

  end_if: //answerfortran

print(Unquoted,"the time spent is ".
expr2text(float((time()-tanalyze)/1000))." s."):

// this ends the routine analyze-solution
```

Appendix 6 : The output for D_{100}, Order 2, on the grid $(-1..1)^3$, optimize=0

----- preparations -----

gridmin = −1, gridmax = 1

se1 = 1, se2 = 0, se3 = 0

order = 2

optimize = 0

check = 0

----- setup equations -----

Discretization of a derivative in 3D with order 2 :
1 times a derivation along the direction e1,
0 times a derivation along the direction e2,
0 times a derivation along the direction e3,
the highest derivative is 1

Establishing the equations for a consistent discretization.

Computing the local derivative error terms.

Equations for optimizing a pure derivative along e1

The equations are ready to be solved.

the time spent is 0.32 s.

----- solve equations -----

Solving the equations for consistent approximations

The solution is written to the file : D_100−O_2−G_−11−OP_0/soln.mu

the time spent is 0.25 s.

----- analyze solution -----

The constraints specified in the file generate-discretization,

gridmin = -1, gridmax = 1,

se1 = 1, se2 = 0, se3 = 0,

order = 2,

optimize = 0,

check = 0,

are satisfied by the following solution, which is written
using the notation k... for a free coefficient
and where T... represents a stencil which is given by a list
of nodes with corresponding weights in the notation {(i,j,k),w}
where the node is given by the indices i,j,k and the weight
by w, and where Dx, Dy and Dz are the grid spacings
in the three coordinate directions,
in the following approximations :
(a line which ends with \ is continued on the next line)

Dx du/dx = Tfx +

kx1 Tx1 + kx2 Tx2 + kx3 Tx3 + kx4 Tx4 + kx5 Tx5 + kx6 Tx6 +

kx7 Tx7 + kx8 Tx8 + kx9 Tx9 + kx10 Tx10 + kx11 Tx11 +

kx12 Tx12 + kx13 Tx13 + kx14 Tx14 + kx15 Tx15 + kx16 Tx16 +

kx17 Tx17, where

Tfx = {(-1,-1,-1),3/2}, {(0,-1,-1),-2}, {(1,-1,-1),1/2}, {(-1,0,-1\
),-1}, {(0,0,-1),1}, {(-1,-1,0),-1}, {(0,-1,0),1};

Tx1 = {(-1,-1,-1),-1}, {(0,-1,-1),2}, {(1,-1,-1),-1}, {(-1,0,-1),1\

```
}, {(0,0,-1),-2}, {(1,0,-1),1};

Tx2 = {(-1,-1,-1),-1}, {(0,-1,-1),1}, {(-1,0,-1),2}, {(0,0,-1),-2}\
, {(-1,1,-1),-1}, {(0,1,-1),1};

Tx3 = {(-1,-1,-1),-3}, {(0,-1,-1),4}, {(1,-1,-1),-1}, {(-1,0,-1),4\
}, {(0,0,-1),-4}, {(-1,1,-1),-1}, {(1,1,-1),1};

Tx4 = {(-1,-1,-1),-1}, {(0,-1,-1),2}, {(1,-1,-1),-1}, {(-1,-1,0),1\
}, {(0,-1,0),-2}, {(1,-1,0),1};

Tx5 = {(-1,-1,-1),-1}, {(0,-1,-1),1}, {(-1,0,-1),1}, {(0,0,-1),-1}\
, {(-1,-1,0),1}, {(0,-1,0),-1}, {(-1,0,0),-1}, {(0,0,0),1};

Tx6 = {(-1,-1,-1),-3}, {(0,-1,-1),4}, {(1,-1,-1),-1}, {(-1,0,-1),2\
}, {(0,0,-1),-2}, {(-1,-1,0),2}, {(0,-1,0),-2}, {(-1,0,0),-1}, {(1\
,0,0),1};

Tx7 = {(-1,-1,-1),-1}, {(-1,0,-1),2}, {(-1,1,-1),-1}, {(-1,-1,0),1\
}, {(-1,0,0),-2}, {(-1,1,0),1};

Tx8 = {(-1,-1,-1),-3}, {(0,-1,-1),2}, {(-1,0,-1),4}, {(0,0,-1),-2}\
, {(-1,1,-1),-1}, {(-1,-1,0),2}, {(0,-1,0),-1}, {(-1,0,0),-2}, {(0\
,1,0),1};

Tx9 = {(-1,-1,-1),-6}, {(0,-1,-1),6}, {(1,-1,-1),-1}, {(-1,0,-1),6\
}, {(0,0,-1),-4}, {(-1,1,-1),-1}, {(-1,-1,0),3}, {(0,-1,0),-2}, {(\
-1,0,0),-2}, {(1,1,0),1};

Tx10 = {(-1,-1,-1),-1}, {(0,-1,-1),1}, {(-1,-1,0),2}, {(0,-1,0),-2\
}, {(-1,-1,1),-1}, {(0,-1,1),1};

Tx11 = {(-1,-1,-1),-3}, {(0,-1,-1),4}, {(1,-1,-1),-1}, {(-1,-1,0),\
4}, {(0,-1,0),-4}, {(-1,-1,1),-1}, {(1,-1,1),1};

Tx12 = {(-1,-1,-1),-1}, {(-1,0,-1),1}, {(-1,-1,0),2}, {(-1,0,0),-2\
}, {(-1,-1,1),-1}, {(-1,0,1),1};

Tx13 = {(-1,-1,-1),-3}, {(0,-1,-1),2}, {(-1,0,-1),2}, {(0,0,-1),-1\
}, {(-1,-1,0),4}, {(0,-1,0),-2}, {(-1,0,0),-2}, {(-1,-1,1),-1}, {(\
```

```
0,0,1),1};

Tx14 = {(-1,-1,-1),-6}, {(0,-1,-1),6}, {(1,-1,-1),-1}, {(-1,0,-1),\
3}, {(0,0,-1),-2}, {(-1,-1,0),6}, {(0,-1,0),-4}, {(-1,0,0),-2}, {(\
-1,-1,1),-1}, {(1,0,1),1};

Tx15 = {(-1,-1,-1),-3}, {(-1,0,-1),4}, {(-1,1,-1),-1}, {(-1,-1,0),\
4}, {(-1,0,0),-4}, {(-1,-1,1),-1}, {(-1,1,1),1};

Tx16 = {(-1,-1,-1),-6}, {(0,-1,-1),3}, {(-1,0,-1),6}, {(0,0,-1),-2\
}, {(-1,1,-1),-1}, {(-1,-1,0),6}, {(0,-1,0),-2}, {(-1,0,0),-4}, {(\
-1,-1,1),-1}, {(0,1,1),1};

Tx17 = {(-1,-1,-1),-10}, {(0,-1,-1),8}, {(1,-1,-1),-1}, {(-1,0,-1)\
,8}, {(0,0,-1),-4}, {(-1,1,-1),-1}, {(-1,-1,0),8}, {(0,-1,0),-4}, \
{(-1,0,0),-4}, {(-1,-1,1),-1}, {(1,1,1),1};

                    Dy du/dy = Tfy +

ky1 Ty1 + ky2 Ty2 + ky3 Ty3 + ky4 Ty4 + ky5 Ty5 + ky6 Ty6 +

    ky7 Ty7 + ky8 Ty8 + ky9 Ty9 + ky10 Ty10 + ky11 Ty11 +

    ky12 Ty12 + ky13 Ty13 + ky14 Ty14 + ky15 Ty15 + ky16 Ty16 +

    ky17 Ty17,  where

Tfy = {(-1,-1,-1),3/2}, {(0,-1,-1),-1}, {(-1,0,-1),-2}, {(0,0,-1),\
1}, {(-1,1,-1),1/2}, {(-1,-1,0),-1}, {(-1,0,0),1};

Ty1 = {(-1,-1,-1),-1}, {(0,-1,-1),2}, {(1,-1,-1),-1}, {(-1,0,-1),1\
}, {(0,0,-1),-2}, {(1,0,-1),1};

Ty2 = {(-1,-1,-1),-1}, {(0,-1,-1),1}, {(-1,0,-1),2}, {(0,0,-1),-2}\
, {(-1,1,-1),-1}, {(0,1,-1),1};

Ty3 = {(-1,-1,-1),-3}, {(0,-1,-1),4}, {(1,-1,-1),-1}, {(-1,0,-1),4\
}, {(0,0,-1),-4}, {(-1,1,-1),-1}, {(1,1,-1),1};

Ty4 = {(-1,-1,-1),-1}, {(0,-1,-1),2}, {(1,-1,-1),-1}, {(-1,-1,0),1\
```

```
}, {(0,-1,0),-2}, {(1,-1,0),1};

Ty5 = {(-1,-1,-1),-1}, {(0,-1,-1),1}, {(-1,0,-1),1}, {(0,0,-1),-1}\
, {(-1,-1,0),1}, {(0,-1,0),-1}, {(-1,0,0),-1}, {(0,0,0),1};

Ty6 = {(-1,-1,-1),-3}, {(0,-1,-1),4}, {(1,-1,-1),-1}, {(-1,0,-1),2\
}, {(0,0,-1),-2}, {(-1,-1,0),2}, {(0,-1,0),-2}, {(-1,0,0),-1}, {(1\
,0,0),1};

Ty7 = {(-1,-1,-1),-1}, {(-1,0,-1),2}, {(-1,1,-1),-1}, {(-1,-1,0),1\
}, {(-1,0,0),-2}, {(-1,1,0),1};

Ty8 = {(-1,-1,-1),-3}, {(0,-1,-1),2}, {(-1,0,-1),4}, {(0,0,-1),-2}\
, {(-1,1,-1),-1}, {(-1,-1,0),2}, {(0,-1,0),-1}, {(-1,0,0),-2}, {(0\
,1,0),1};

Ty9 = {(-1,-1,-1),-6}, {(0,-1,-1),6}, {(1,-1,-1),-1}, {(-1,0,-1),6\
}, {(0,0,-1),-4}, {(-1,1,-1),-1}, {(-1,-1,0),3}, {(0,-1,0),-2}, {(\
-1,0,0),-2}, {(1,1,0),1};

Ty10 = {(-1,-1,-1),-1}, {(0,-1,-1),1}, {(-1,-1,0),2}, {(0,-1,0),-2\
}, {(-1,-1,1),-1}, {(0,-1,1),1};

Ty11 = {(-1,-1,-1),-3}, {(0,-1,-1),4}, {(1,-1,-1),-1}, {(-1,-1,0),\
4}, {(0,-1,0),-4}, {(-1,-1,1),-1}, {(1,-1,1),1};

Ty12 = {(-1,-1,-1),-1}, {(-1,0,-1),1}, {(-1,-1,0),2}, {(-1,0,0),-2\
}, {(-1,-1,1),-1}, {(-1,0,1),1};

Ty13 = {(-1,-1,-1),-3}, {(0,-1,-1),2}, {(-1,0,-1),2}, {(0,0,-1),-1\
}, {(-1,-1,0),4}, {(0,-1,0),-2}, {(-1,0,0),-2}, {(-1,-1,1),-1}, {(\
0,0,1),1};

Ty14 = {(-1,-1,-1),-6}, {(0,-1,-1),6}, {(1,-1,-1),-1}, {(-1,0,-1),\
3}, {(0,0,-1),-2}, {(-1,-1,0),6}, {(0,-1,0),-4}, {(-1,0,0),-2}, {(\
-1,-1,1),-1}, {(1,0,1),1};

Ty15 = {(-1,-1,-1),-3}, {(-1,0,-1),4}, {(-1,1,-1),-1}, {(-1,-1,0),\
4}, {(-1,0,0),-4}, {(-1,-1,1),-1}, {(-1,1,1),1};
```

```
Ty16 = {(-1,-1,-1),-6}, {(0,-1,-1),3}, {(-1,0,-1),6}, {(0,0,-1),-2\
}, {(-1,1,-1),-1}, {(-1,-1,0),6}, {(0,-1,0),-2}, {(-1,0,0),-4}, {(\
-1,-1,1),-1}, {(0,1,1),1};

Ty17 = {(-1,-1,-1),-10}, {(0,-1,-1),8}, {(1,-1,-1),-1}, {(-1,0,-1)\
,8}, {(0,0,-1),-4}, {(-1,1,-1),-1}, {(-1,-1,0),8}, {(0,-1,0),-4}, \
{(-1,0,0),-4}, {(-1,-1,1),-1}, {(1,1,1),1};

                    Dz du/dz = Tfz +

kz1 Tz1 + kz2 Tz2 + kz3 Tz3 + kz4 Tz4 + kz5 Tz5 + kz6 Tz6 +

   kz7 Tz7 + kz8 Tz8 + kz9 Tz9 + kz10 Tz10 + kz11 Tz11 +

   kz12 Tz12 + kz13 Tz13 + kz14 Tz14 + kz15 Tz15 + kz16 Tz16 +

   kz17 Tz17,   where

Tfz = {(-1,-1,-1),3/2}, {(0,-1,-1),-1}, {(-1,0,-1),-1}, {(-1,-1,0)\
,-2}, {(0,-1,0),1}, {(-1,0,0),1}, {(-1,-1,1),1/2};

Tz1 = {(-1,-1,-1),-1}, {(0,-1,-1),2}, {(1,-1,-1),-1}, {(-1,0,-1),1\
}, {(0,0,-1),-2}, {(1,0,-1),1};

Tz2 = {(-1,-1,-1),-1}, {(0,-1,-1),1}, {(-1,0,-1),2}, {(0,0,-1),-2}\
, {(-1,1,-1),-1}, {(0,1,-1),1};

Tz3 = {(-1,-1,-1),-3}, {(0,-1,-1),4}, {(1,-1,-1),-1}, {(-1,0,-1),4\
}, {(0,0,-1),-4}, {(-1,1,-1),-1}, {(1,1,-1),1};

Tz4 = {(-1,-1,-1),-1}, {(0,-1,-1),2}, {(1,-1,-1),-1}, {(-1,-1,0),1\
}, {(0,-1,0),-2}, {(1,-1,0),1};

Tz5 = {(-1,-1,-1),-1}, {(0,-1,-1),1}, {(-1,0,-1),1}, {(0,0,-1),-1}\
, {(-1,-1,0),1}, {(0,-1,0),-1}, {(-1,0,0),-1}, {(0,0,0),1};

Tz6 = {(-1,-1,-1),-3}, {(0,-1,-1),4}, {(1,-1,-1),-1}, {(-1,0,-1),2\
}, {(0,0,-1),-2}, {(-1,-1,0),2}, {(0,-1,0),-2}, {(-1,0,0),-1}, {(1\
,0,0),1};
```

```
Tz7 = {(-1,-1,-1),-1}, {(-1,0,-1),2}, {(-1,1,-1),-1}, {(-1,-1,0),1\
}, {(-1,0,0),-2}, {(-1,1,0),1};

Tz8 = {(-1,-1,-1),-3}, {(0,-1,-1),2}, {(-1,0,-1),4}, {(0,0,-1),-2}\
, {(-1,1,-1),-1}, {(-1,-1,0),2}, {(0,-1,0),-1}, {(-1,0,0),-2}, {(0\
,1,0),1};

Tz9 = {(-1,-1,-1),-6}, {(0,-1,-1),6}, {(1,-1,-1),-1}, {(-1,0,-1),6\
}, {(0,0,-1),-4}, {(-1,1,-1),-1}, {(-1,-1,0),3}, {(0,-1,0),-2}, {(\
-1,0,0),-2}, {(1,1,0),1};

Tz10 = {(-1,-1,-1),-1}, {(0,-1,-1),1}, {(-1,-1,0),2}, {(0,-1,0),-2\
}, {(-1,-1,1),-1}, {(0,-1,1),1};

Tz11 = {(-1,-1,-1),-3}, {(0,-1,-1),4}, {(1,-1,-1),-1}, {(-1,-1,0),\
4}, {(0,-1,0),-4}, {(-1,-1,1),-1}, {(1,-1,1),1};

Tz12 = {(-1,-1,-1),-1}, {(-1,0,-1),1}, {(-1,-1,0),2}, {(-1,0,0),-2\
}, {(-1,-1,1),-1}, {(-1,0,1),1};

Tz13 = {(-1,-1,-1),-3}, {(0,-1,-1),2}, {(-1,0,-1),2}, {(0,0,-1),-1\
}, {(-1,-1,0),4}, {(0,-1,0),-2}, {(-1,0,0),-2}, {(-1,-1,1),-1}, {(\
0,0,1),1};

Tz14 = {(-1,-1,-1),-6}, {(0,-1,-1),6}, {(1,-1,-1),-1}, {(-1,0,-1),\
3}, {(0,0,-1),-2}, {(-1,-1,0),6}, {(0,-1,0),-4}, {(-1,0,0),-2}, {(\
-1,-1,1),-1}, {(1,0,1),1};

Tz15 = {(-1,-1,-1),-3}, {(-1,0,-1),4}, {(-1,1,-1),-1}, {(-1,-1,0),\
4}, {(-1,0,0),-4}, {(-1,-1,1),-1}, {(-1,1,1),1};

Tz16 = {(-1,-1,-1),-6}, {(0,-1,-1),3}, {(-1,0,-1),6}, {(0,0,-1),-2\
}, {(-1,1,-1),-1}, {(-1,-1,0),6}, {(0,-1,0),-2}, {(-1,0,0),-4}, {(\
-1,-1,1),-1}, {(0,1,1),1};

Tz17 = {(-1,-1,-1),-10}, {(0,-1,-1),8}, {(1,-1,-1),-1}, {(-1,0,-1)\
,8}, {(0,0,-1),-4}, {(-1,1,-1),-1}, {(-1,-1,0),8}, {(0,-1,0),-4}, \
{(-1,0,0),-4}, {(-1,-1,1),-1}, {(1,1,1),1};
```

The grid-aligned one-dimensional solution which is excluded

from the patent is given by

```
kx1 = 0, kx2 = 0, kx3 = 0, kx4 = 0, kx5 = 0, kx6 = 1/2, kx7 = 0, k\
x8 = 0, kx9 = 0, kx10 = 0, kx11 = 0, kx12 = 0, kx13 = 0, kx14 = 0,\
 kx15 = 0, kx16 = 0, kx17 = 0
```

and

```
ky1 = 0, ky2 = 0, ky3 = 0, ky4 = 0, ky5 = 0, ky6 = 0, ky7 = 0, ky8\
 = 1/2, ky9 = 0, ky10 = 0, ky11 = 0, ky12 = 0, ky13 = 0, ky14 = 0,\
 ky15 = 0, ky16 = 0, ky17 = 0
```

and

```
kz1 = 0, kz2 = 0, kz3 = 0, kz4 = 0, kz5 = 0, kz6 = 0, kz7 = 0, kz8\
 = 0, kz9 = 0, kz10 = 0, kz11 = 0, kz12 = 0, kz13 = 1/2, kz14 = 0,\
 kz15 = 0, kz16 = 0, kz17 = 0
```

APPENDIX 7 : THE OUTPUT FOR D_{100}, ORDER 2, ON THE GRID $(-1..1)^3$, OPTIMIZE=1

The first part of this output is the same as of Appendix 6. It is ended by :

...

```
Tz16 = {(-1,-1,-1),-6}, {(0,-1,-1),3}, {(-1,0,-1),6}, {(0,0,-1),-2\
}, {(-1,1,-1),-1}, {(-1,-1,0),6}, {(0,-1,0),-2}, {(-1,0,0),-4}, {(\
-1,-1,1),-1}, {(0,1,1),1};

Tz17 = {(-1,-1,-1),-10}, {(0,-1,-1),8}, {(1,-1,-1),-1}, {(-1,0,-1)\
,8}, {(0,0,-1),-4}, {(-1,1,-1),-1}, {(-1,-1,0),8}, {(0,-1,0),-4}, \
{(-1,0,0),-4}, {(-1,-1,1),-1}, {(1,1,1),1};

    The grid-aligned one-dimensional solution which is excluded
    from the patent is given by

kx1 = 0, kx2 = 0, kx3 = 0, kx4 = 0, kx5 = 0, kx6 = 1/2, kx7 = 0, k\
x8 = 0, kx9 = 0, kx10 = 0, kx11 = 0, kx12 = 0, kx13 = 0, kx14 = 0,\
 kx15 = 0, kx16 = 0, kx17 = 0
```

and

```
ky1 = 0, ky2 = 0, ky3 = 0, ky4 = 0, ky5 = 0, ky6 = 0, ky7 = 0, ky8\
 = 1/2, ky9 = 0, ky10 = 0, ky11 = 0, ky12 = 0, ky13 = 0, ky14 = 0,\
 ky15 = 0, ky16 = 0, ky17 = 0
```

and

```
kz1 = 0, kz2 = 0, kz3 = 0, kz4 = 0, kz5 = 0, kz6 = 0, kz7 = 0, kz8\
 = 0, kz9 = 0, kz10 = 0, kz11 = 0, kz12 = 0, kz13 = 1/2, kz14 = 0,\
 kz15 = 0, kz16 = 0, kz17 = 0

 We use the notation
 a for the angle alpha, the rotation around the grid z-axis,
 b for the angle beta,  the rotation around the new y'-axis,
 c for the angle gamma, the rotation around newest x''-axis,
```

which is the axis e1 of the local basis B, as explained in the patent application.
The present optimization imposes constraints on the error terms.
The error terms in the local basis are written to the files direrror.mu and direrror.txt.
The notation Epqr means the error in the approximation related to the mixed pth derivative w.r.t. direction e1,
the qth derivative w.r.t. to e2 and the rth derivative w.r.t. e3
e.g. E201 is the contribution related to the mixed 2nd derivative along axis e1, and the first derivative along axis e3

The error term E003 = 0

The error term E012 = 0

The error term E021 = 0

The error term E030 = 0

The error terms $E003$ etc. are stored upon request in separate files. As an example, the program generates the file `D_100-0_2-G_-11-OP_1/direrror003.mu` which contains just this error, which is assigned to the variable direrrtmp. The file contains some 439 lines, starting with

```
sysassign(direrrtmp, hold(Dx*Dy*cos(a)*cos(b)*cos(c)*sin(a)*sin(b)\
*sin(c)^2 - Dx*Dy*kz5*cos(a)*cos(b)*cos(c)*sin(a)*sin(b)*sin(c)^2 \
- 2*Dx*Dy*kz6*cos(a)*cos(b)*cos(c)*sin(a)*sin(b)*sin(c)^2 - 2*Dx*D\
y*kz8*cos(a)*cos(b)*cos(c)*sin(a)*sin(b)*sin(c)^2 - 4*Dx*Dy*kz9*co\
s(a)*cos(b)*cos(c)*sin(a)*sin(b)*sin(c)^2 - 2*Dx*Dy*kz13*cos(a)*co\
```

... and ending with ...

```
2*sin(b)^2 + Dx^2/Dy*Dz*ky9*cos(a)^2*cos(b)^2*cos(c)^3*sin(a)*sin(\
b)^2 + 2*Dx^2/Dy*Dz*ky11*cos(a)^2*cos(b)^2*cos(c)^3*sin(a)*sin(b)^\
2 + 2/Dx*Dy^2*Dz*kx15*cos(a)*cos(b)^2*cos(c)^3*sin(a)^2*sin(b)^2 +\
 2*Dx^2/Dy*Dz*ky14*cos(a)^2*cos(b)^2*cos(c)^3*sin(a)*sin(b)^2 + 2/\
Dx*Dy^2*Dz*kx16*cos(a)*cos(b)^2*cos(c)^3*sin(a)^2*sin(b)^2 + 2/Dx*\
Dy^2*Dz*kx17*cos(a)*cos(b)^2*cos(c)^3*sin(a)^2*sin(b)^2 + 2*Dx^2/D\
y*Dz*ky17*cos(a)^2*cos(b)^2*cos(c)^3*sin(a)*sin(b)^2)):
```

APPENDIX 8 : THE OUTPUT FOR D_{100}, ORDER 2, ON THE GRID $(-1..1)^3$, OPTIMIZE=2

```
----- preparations ------

gridmin = -1, gridmax = 1

se1 = 1, se2 = 0, se3 = 0

order = 2

optimize = 2

check = 0

------ setup equations ------

Discretization of a derivative in 3D with order 2 :
1 times a derivation along the direction e1,
0 times a derivation along the direction e2,
0 times a derivation along the direction e3,
the highest derivative is 1

Establishing the equations for a consistent discretization.

Computing the local derivative error terms.

Equations for optimizing a pure derivative along e1

The equations are ready to be solved.

the time spent is 0.3 s.

----- solve equations ------

Solving the equations for consistent approximations

Solving the directional equations - along x1
```

Solving the directional equations – along diag x1–x2

Solving the directional equations – along diag x1–x2–x3

The solution is written to the file : D_100–O_2–G_–11–OP_2/soln.mu

the time spent is 31.4 s.

––––– analyze solution –––––

The constraints specified in the file generate–discretization,

gridmin = –1, gridmax = 1,

se1 = 1, se2 = 0, se3 = 0,

order = 2,

optimize = 2,

check = 0,

are satisfied by the following solution, which is written using the notation k... for a free coefficient and where T... represents a stencil which is given by a list of nodes with corresponding weights in the notation {(i,j,k),w} where the node is given by the indices i,j,k and the weight by w, and where Dx, Dy and Dz are the grid spacings in the three coordinate directions, in the following approximations :
(a line which ends with \ is continued on the next line)

Dx du/dx = Tfx +

kx1 Tx1 + kx2 Tx2 + kx3 Tx3 + kx4 Tx4 + kx5 Tx5 + kx6 Tx6 +

kx7 Tx7 + kx8 Tx8 + kx9 Tx9 + kx10 Tx10, where

Tfx = {(–1,–1,–1),–1/2}, {(0,–1,–1),1}, {(1,–1,–1),–1/2}, {(–1,0,–\
1),1/2}, {(0,0,–1),–1}, {(1,0,–1),1/2}, {(–1,–1,0),1/2}, {(0,–1,0)\

```
,-1}, {(1,-1,0),1/2}, {(-1,0,0),-1}, {(0,0,0),1};

Tx1 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-2\
}, {(0,0,-1),4}, {(1,0,-1),-2}, {(-1,1,-1),1}, {(0,1,-1),-2}, {(1,\
1,-1),1};

Tx2 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-1\
}, {(0,0,-1),2}, {(1,0,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),2}, {(1,\
-1,0),-1}, {(-1,0,0),1}, {(0,0,0),-2}, {(1,0,0),1};

Tx3 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-2}, {(0,0,-1),2}\
, {(-1,1,-1),1}, {(0,1,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),1}, {(-1\
,0,0),2}, {(0,0,0),-2}, {(-1,1,0),-1}, {(0,1,0),1};

Tx4 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-6\
}, {(0,0,-1),8}, {(1,0,-1),-2}, {(-1,1,-1),2}, {(0,1,-1),-2}, {(-1\
,-1,0),-3}, {(0,-1,0),4}, {(1,-1,0),-1}, {(-1,0,0),4}, {(0,0,0),-4\
}, {(-1,1,0),-1}, {(1,1,0),1};

Tx5 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,-1,0),-2\
}, {(0,-1,0),4}, {(1,-1,0),-2}, {(-1,-1,1),1}, {(0,-1,1),-2}, {(1,\
-1,1),1};

Tx6 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-1}, {(0,0,-1),1}\
, {(-1,-1,0),-2}, {(0,-1,0),2}, {(-1,0,0),2}, {(0,0,0),-2}, {(-1,-\
1,1),1}, {(0,-1,1),-1}, {(-1,0,1),-1}, {(0,0,1),1};

Tx7 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-3\
}, {(0,0,-1),4}, {(1,0,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),8}, {(1,\
-1,0),-2}, {(-1,0,0),4}, {(0,0,0),-4}, {(-1,-1,1),2}, {(0,-1,1),-2\
}, {(-1,0,1),-1}, {(1,0,1),1};

Tx8 = {(-1,-1,-1),1}, {(-1,0,-1),-2}, {(-1,1,-1),1}, {(-1,-1,0),-2\
}, {(-1,0,0),4}, {(-1,1,0),-2}, {(-1,-1,1),1}, {(-1,0,1),-2}, {(-1\
,1,1),1};

Tx9 = {(-1,-1,-1),4}, {(0,-1,-1),-3}, {(-1,0,-1),-6}, {(0,0,-1),4}\
, {(-1,1,-1),2}, {(0,1,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),4}, {(-1\
,0,0),8}, {(0,0,0),-4}, {(-1,1,0),-2}, {(-1,-1,1),2}, {(0,-1,1),-1\
}, {(-1,0,1),-2}, {(0,1,1),1};
```

```
Tx10 = {(-1,-1,-1),10}, {(0,-1,-1),-12}, {(1,-1,-1),3}, {(-1,0,-1)\
,-12}, {(0,0,-1),12}, {(1,0,-1),-2}, {(-1,1,-1),3}, {(0,1,-1),-2},\
 {(-1,-1,0),-12}, {(0,-1,0),12}, {(1,-1,0),-2}, {(-1,0,0),12}, {(0\
,0,0),-8}, {(-1,1,0),-2}, {(-1,-1,1),3}, {(0,-1,1),-2}, {(-1,0,1),\
-2}, {(1,1,1),1};
```

Dy du/dy = Tfy +

ky1 Ty1 + ky2 Ty2 + ky3 Ty3 + ky4 Ty4 + ky5 Ty5 + ky6 Ty6 +

ky7 Ty7 + ky8 Ty8 + ky9 Ty9 + ky10 Ty10, where

```
Tfy = {(-1,-1,-1),-3/2}, {(0,-1,-1),2}, {(1,-1,-1),-1/2}, {(-1,0,-\
1),5/2}, {(0,0,-1),-3}, {(1,0,-1),1/2}, {(-1,1,-1),-1}, {(0,1,-1),\
1}, {(-1,-1,0),1/2}, {(0,-1,0),-1}, {(-1,0,0),-1}, {(0,0,0),1}, {(\
-1,1,0),1/2};

Ty1 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-2\
}, {(0,0,-1),4}, {(1,0,-1),-2}, {(-1,1,-1),1}, {(0,1,-1),-2}, {(1,\
1,-1),1};

Ty2 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-1\
}, {(0,0,-1),2}, {(1,0,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),2}, {(1,\
-1,0),-1}, {(-1,0,0),1}, {(0,0,0),-2}, {(1,0,0),1};

Ty3 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-2}, {(0,0,-1),2}\
, {(-1,1,-1),1}, {(0,1,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),1}, {(-1\
,0,0),2}, {(0,0,0),-2}, {(-1,1,0),-1}, {(0,1,0),1};

Ty4 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-6\
}, {(0,0,-1),8}, {(1,0,-1),-2}, {(-1,1,-1),2}, {(0,1,-1),-2}, {(-1\
,-1,0),-3}, {(0,-1,0),4}, {(1,-1,0),-1}, {(-1,0,0),4}, {(0,0,0),-4\
}, {(-1,1,0),-1}, {(1,1,0),1};

Ty5 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,-1,0),-2\
}, {(0,-1,0),4}, {(1,-1,0),-2}, {(-1,-1,1),1}, {(0,-1,1),-2}, {(1,\
-1,1),1};

Ty6 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-1}, {(0,0,-1),1}\
```

```
, {(-1,-1,0),-2}, {(0,-1,0),2}, {(-1,0,0),2}, {(0,0,0),-2}, {(-1,-\
1,1),1}, {(0,-1,1),-1}, {(-1,0,1),-1}, {(0,0,1),1};

Ty7 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-3\
}, {(0,0,-1),4}, {(1,0,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),8}, {(1,\
-1,0),-2}, {(-1,0,0),4}, {(0,0,0),-4}, {(-1,-1,1),2}, {(0,-1,1),-2\
}, {(-1,0,1),-1}, {(1,0,1),1};

Ty8 = {(-1,-1,-1),1}, {(-1,0,-1),-2}, {(-1,1,-1),1}, {(-1,-1,0),-2\
}, {(-1,0,0),4}, {(-1,1,0),-2}, {(-1,-1,1),1}, {(-1,0,1),-2}, {(-1\
,1,1),1};

Ty9 = {(-1,-1,-1),4}, {(0,-1,-1),-3}, {(-1,0,-1),-6}, {(0,0,-1),4}\
, {(-1,1,-1),2}, {(0,1,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),4}, {(-1\
,0,0),8}, {(0,0,0),-4}, {(-1,1,0),-2}, {(-1,-1,1),2}, {(0,-1,1),-1\
}, {(-1,0,1),-2}, {(0,1,1),1};

Ty10 = {(-1,-1,-1),10}, {(0,-1,-1),-12}, {(1,-1,-1),3}, {(-1,0,-1)\
,-12}, {(0,0,-1),12}, {(1,0,-1),-2}, {(-1,1,-1),3}, {(0,1,-1),-2},\
 {(-1,-1,0),-12}, {(0,-1,0),12}, {(1,-1,0),-2}, {(-1,0,0),12}, {(0\
,0,0),-8}, {(-1,1,0),-2}, {(-1,-1,1),3}, {(0,-1,1),-2}, {(-1,0,1),\
-2}, {(1,1,1),1};

                    Dz du/dz = Tfz +

kz1 Tz1 + kz2 Tz2 + kz3 Tz3 + kz4 Tz4 + kz5 Tz5 + kz6 Tz6 +

   kz7 Tz7 + kz8 Tz8 + kz9 Tz9 + kz10 Tz10,  where

Tfz = {(-1,-1,-1),-7/2}, {(0,-1,-1),3}, {(1,-1,-1),-1/2}, {(-1,0,-\
1),3}, {(0,0,-1),-2}, {(-1,1,-1),-1/2}, {(-1,-1,0),5}, {(0,-1,0),-\
4}, {(1,-1,0),1/2}, {(-1,0,0),-4}, {(0,0,0),2}, {(-1,1,0),1/2}, {(\
-1,-1,1),-3/2}, {(0,-1,1),1}, {(-1,0,1),1};

Tz1 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-2\
}, {(0,0,-1),4}, {(1,0,-1),-2}, {(-1,1,-1),1}, {(0,1,-1),-2}, {(1,\
1,-1),1};

Tz2 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-1\
}, {(0,0,-1),2}, {(1,0,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),2}, {(1,\
```

```
-1,0),-1}, {(-1,0,0),1}, {(0,0,0),-2}, {(1,0,0),1};

Tz3 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-2}, {(0,0,-1),2}\
, {(-1,1,-1),1}, {(0,1,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),1}, {(-1\
,0,0),2}, {(0,0,0),-2}, {(-1,1,0),-1}, {(0,1,0),1};

Tz4 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-6\
}, {(0,0,-1),8}, {(1,0,-1),-2}, {(-1,1,-1),2}, {(0,1,-1),-2}, {(-1\
,-1,0),-3}, {(0,-1,0),4}, {(1,-1,0),-1}, {(-1,0,0),4}, {(0,0,0),-4\
}, {(-1,1,0),-1}, {(1,1,0),1};

Tz5 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,-1,0),-2\
}, {(0,-1,0),4}, {(1,-1,0),-2}, {(-1,-1,1),1}, {(0,-1,1),-2}, {(1,\
-1,1),1};

Tz6 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-1}, {(0,0,-1),1}\
, {(-1,-1,0),-2}, {(0,-1,0),2}, {(-1,0,0),2}, {(0,0,0),-2}, {(-1,-\
1,1),1}, {(0,-1,1),-1}, {(-1,0,1),-1}, {(0,0,1),1};

Tz7 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-3\
}, {(0,0,-1),4}, {(1,0,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),8}, {(1,\
-1,0),-2}, {(-1,0,0),4}, {(0,0,0),-4}, {(-1,-1,1),2}, {(0,-1,1),-2\
}, {(-1,0,1),-1}, {(1,0,1),1};

Tz8 = {(-1,-1,-1),1}, {(-1,0,-1),-2}, {(-1,1,-1),1}, {(-1,-1,0),-2\
}, {(-1,0,0),4}, {(-1,1,0),-2}, {(-1,-1,1),1}, {(-1,0,1),-2}, {(-1\
,1,1),1};

Tz9 = {(-1,-1,-1),4}, {(0,-1,-1),-3}, {(-1,0,-1),-6}, {(0,0,-1),4}\
, {(-1,1,-1),2}, {(0,1,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),4}, {(-1\
,0,0),8}, {(0,0,0),-4}, {(-1,1,0),-2}, {(-1,-1,1),2}, {(0,-1,1),-1\
}, {(-1,0,1),-2}, {(0,1,1),1};

Tz10 = {(-1,-1,-1),10}, {(0,-1,-1),-12}, {(1,-1,-1),3}, {(-1,0,-1)\
,-12}, {(0,0,-1),12}, {(1,0,-1),-2}, {(-1,1,-1),3}, {(0,1,-1),-2},\
 {(-1,-1,0),-12}, {(0,-1,0),12}, {(1,-1,0),-2}, {(-1,0,0),12}, {(0\
,0,0),-8}, {(-1,1,0),-2}, {(-1,-1,1),3}, {(0,-1,1),-2}, {(-1,0,1),\
-2}, {(1,1,1),1};
```

APPENDIX 9 : THE OUTPUT FOR D_{100}, ORDER 4, ON THE GRID $(-2..2)^3$, OPTIMIZE=0

----- preparations -----

gridmin = −2, gridmax = 2

se1 = 1, se2 = 0, se3 = 0

order = 4

optimize = 0

check = 0

----- setup equations -----

Discretization of a derivative in 3D with order 4 :
1 times a derivation along the direction e1,
0 times a derivation along the direction e2,
0 times a derivation along the direction e3,
the highest derivative is 1

Establishing the equations for a consistent discretization.

Computing the local derivative error terms.

Equations for optimizing a pure derivative along e1

The equations are ready to be solved.

the time spent is 5.55 s.

----- solve equations -----

Solving the equations for consistent approximations

The solution is written to the file : D_100–O_4–G_–22–OP_0/soln.mu

the time spent is 2.57 s.

----- analyze solution -----

The constraints specified in the file generate-discretization,

gridmin = -2, gridmax = 2,

se1 = 1, se2 = 0, se3 = 0,

order = 4,

optimize = 0,

check = 0,

are satisfied by the following solution, which is written
using the notation k... for a free coefficient
and where T... represents a stencil which is given by a list
of nodes with corresponding weights in the notation {(i,j,k),w}
where the node is given by the indices i,j,k and the weight
by w, and where Dx, Dy and Dz are the grid spacings
in the three coordinate directions,
in the following approximations :
(a line which ends with \ is continued on the next line)

Dx du/dx = Tfx +

kx1 Tx1 + kx2 Tx2 + kx3 Tx3 + kx4 Tx4 + kx5 Tx5 + kx6 Tx6 +

kx7 Tx7 + kx8 Tx8 + kx9 Tx9 + kx10 Tx10 + kx11 Tx11 +

kx12 Tx12 + kx13 Tx13 + kx14 Tx14 + kx15 Tx15 + kx16 Tx16 +

kx17 Tx17 + kx18 Tx18 + kx19 Tx19 + kx20 Tx20 + kx21 Tx21 +

kx22 Tx22 + kx23 Tx23 + kx24 Tx24 + kx25 Tx25 + kx26 Tx26 +

kx27 Tx27 + kx28 Tx28 + kx29 Tx29 + kx30 Tx30 + kx31 Tx31 +

```
kx32 Tx32 + kx33 Tx33 + kx34 Tx34 + kx35 Tx35 + kx36 Tx36 +
kx37 Tx37 + kx38 Tx38 + kx39 Tx39 + kx40 Tx40 + kx41 Tx41 +
kx42 Tx42 + kx43 Tx43 + kx44 Tx44 + kx45 Tx45 + kx46 Tx46 +
kx47 Tx47 + kx48 Tx48 + kx49 Tx49 + kx50 Tx50 + kx51 Tx51 +
kx52 Tx52 + kx53 Tx53 + kx54 Tx54 + kx55 Tx55 + kx56 Tx56 +
kx57 Tx57 + kx58 Tx58 + kx59 Tx59 + kx60 Tx60 + kx61 Tx61 +
kx62 Tx62 + kx63 Tx63 + kx64 Tx64 + kx65 Tx65 + kx66 Tx66 +
kx67 Tx67 + kx68 Tx68 + kx69 Tx69 + kx70 Tx70 + kx71 Tx71 +
kx72 Tx72 + kx73 Tx73 + kx74 Tx74 + kx75 Tx75 + kx76 Tx76 +
kx77 Tx77 + kx78 Tx78 + kx79 Tx79 + kx80 Tx80 + kx81 Tx81 +
kx82 Tx82 + kx83 Tx83 + kx84 Tx84 + kx85 Tx85 + kx86 Tx86 +
kx87 Tx87 + kx88 Tx88 + kx89 Tx89 + kx90 Tx90,  where

Tfx = {(-2,-2,-2),77/12}, {(-1,-2,-2),-38/3}, {(0,-2,-2),7}, {(1,-\
2,-2),-2/3}, {(2,-2,-2),-1/12}, {(-2,-1,-2),-26/3}, {(-1,-1,-2),16\
}, {(0,-1,-2),-8}, {(1,-1,-2),2/3}, {(-2,0,-2),5/2}, {(-1,0,-2),-4\
}, {(0,0,-2),3/2}, {(-2,-2,-1),-26/3}, {(-1,-2,-1),16}, {(0,-2,-1)\
,-8}, {(1,-2,-1),2/3}, {(-2,-1,-1),10}, {(-1,-1,-1),-16}, {(0,-1,-\
1),6}, {(-2,0,-1),-2}, {(-1,0,-1),2}, {(-2,-2,0),5/2}, {(-1,-2,0),\
-4}, {(0,-2,0),3/2}, {(-2,-1,0),-2}, {(-1,-1,0),2};

Tx1 = {(-2,-2,-2),-1}, {(-1,-2,-2),4}, {(0,-2,-2),-6}, {(1,-2,-2),\
4}, {(2,-2,-2),-1}, {(-2,-1,-2),1}, {(-1,-1,-2),-4}, {(0,-1,-2),6}\
, {(1,-1,-2),-4}, {(2,-1,-2),1};

Tx2 = {(-2,-2,-2),-1}, {(-1,-2,-2),3}, {(0,-2,-2),-3}, {(1,-2,-2),\
1}, {(-2,-1,-2),2}, {(-1,-1,-2),-6}, {(0,-1,-2),6}, {(1,-1,-2),-2}\
, {(-2,0,-2),-1}, {(-1,0,-2),3}, {(0,0,-2),-3}, {(1,0,-2),1};
```

```
Tx3 = {(-2,-2,-2),-5}, {(-1,-2,-2),16}, {(0,-2,-2),-18}, {(1,-2,-2\
),8}, {(2,-2,-2),-1}, {(-2,-1,-2),8}, {(-1,-1,-2),-24}, {(0,-1,-2)\
,24}, {(1,-1,-2),-8}, {(-2,0,-2),-3}, {(-1,0,-2),8}, {(0,0,-2),-6}\
, {(2,0,-2),1};

Tx4 = {(-2,-2,-2),-1}, {(-1,-2,-2),2}, {(0,-2,-2),-1}, {(-2,-1,-2)\
,3}, {(-1,-1,-2),-6}, {(0,-1,-2),3}, {(-2,0,-2),-3}, {(-1,0,-2),6}\
, {(0,0,-2),-3}, {(-2,1,-2),1}, {(-1,1,-2),-2}, {(0,1,-2),1};

Tx5 = {(-2,-2,-2),-5}, {(-1,-2,-2),12}, {(0,-2,-2),-9}, {(1,-2,-2)\
,2}, {(-2,-1,-2),12}, {(-1,-1,-2),-27}, {(0,-1,-2),18}, {(1,-1,-2)\
,-3}, {(-2,0,-2),-9}, {(-1,0,-2),18}, {(0,0,-2),-9}, {(-2,1,-2),2}\
, {(-1,1,-2),-3}, {(1,1,-2),1};

Tx6 = {(-2,-2,-2),-15}, {(-1,-2,-2),40}, {(0,-2,-2),-36}, {(1,-2,-\
2),12}, {(2,-2,-2),-1}, {(-2,-1,-2),30}, {(-1,-1,-2),-72}, {(0,-1,\
-2),54}, {(1,-1,-2),-12}, {(-2,0,-2),-18}, {(-1,0,-2),36}, {(0,0,-\
2),-18}, {(-2,1,-2),3}, {(-1,1,-2),-4}, {(2,1,-2),1};

Tx7 = {(-2,-2,-2),-1}, {(-1,-2,-2),1}, {(-2,-1,-2),4}, {(-1,-1,-2)\
,-4}, {(-2,0,-2),-6}, {(-1,0,-2),6}, {(-2,1,-2),4}, {(-1,1,-2),-4}\
, {(-2,2,-2),-1}, {(-1,2,-2),1};

Tx8 = {(-2,-2,-2),-5}, {(-1,-2,-2),8}, {(0,-2,-2),-3}, {(-2,-1,-2)\
,16}, {(-1,-1,-2),-24}, {(0,-1,-2),8}, {(-2,0,-2),-18}, {(-1,0,-2)\
,24}, {(0,0,-2),-6}, {(-2,1,-2),8}, {(-1,1,-2),-8}, {(-2,2,-2),-1}\
, {(0,2,-2),1};

Tx9 = {(-2,-2,-2),-15}, {(-1,-2,-2),30}, {(0,-2,-2),-18}, {(1,-2,-\
2),3}, {(-2,-1,-2),40}, {(-1,-1,-2),-72}, {(0,-1,-2),36}, {(1,-1,-\
2),-4}, {(-2,0,-2),-36}, {(-1,0,-2),54}, {(0,0,-2),-18}, {(-2,1,-2\
),12}, {(-1,1,-2),-12}, {(-2,2,-2),-1}, {(1,2,-2),1};

Tx10 = {(-2,-2,-2),-35}, {(-1,-2,-2),80}, {(0,-2,-2),-60}, {(1,-2,\
-2),16}, {(2,-2,-2),-1}, {(-2,-1,-2),80}, {(-1,-1,-2),-160}, {(0,-\
1,-2),96}, {(1,-1,-2),-16}, {(-2,0,-2),-60}, {(-1,0,-2),96}, {(0,0\
,-2),-36}, {(-2,1,-2),16}, {(-1,1,-2),-16}, {(-2,2,-2),-1}, {(2,2,\
-2),1};

Tx11 = {(-2,-2,-2),-1}, {(-1,-2,-2),4}, {(0,-2,-2),-6}, {(1,-2,-2)\
```

```
,4}, {(2,-2,-2),-1}, {(-2,-2,-1),1}, {(-1,-2,-1),-4}, {(0,-2,-1),6\
}, {(1,-2,-1),-4}, {(2,-2,-1),1};

Tx12 = {(-2,-2,-2),-1}, {(-1,-2,-2),3}, {(0,-2,-2),-3}, {(1,-2,-2)\
,1}, {(-2,-1,-2),1}, {(-1,-1,-2),-3}, {(0,-1,-2),3}, {(1,-1,-2),-1\
}, {(-2,-2,-1),1}, {(-1,-2,-1),-3}, {(0,-2,-1),3}, {(1,-2,-1),-1},\
 {(-2,-1,-1),-1}, {(-1,-1,-1),3}, {(0,-1,-1),-3}, {(1,-1,-1),1};

Tx13 = {(-2,-2,-2),-5}, {(-1,-2,-2),16}, {(0,-2,-2),-18}, {(1,-2,-\
2),8}, {(2,-2,-2),-1}, {(-2,-1,-2),4}, {(-1,-1,-2),-12}, {(0,-1,-2\
),12}, {(1,-1,-2),-4}, {(-2,-2,-1),4}, {(-1,-2,-1),-12}, {(0,-2,-1\
),12}, {(1,-2,-1),-4}, {(-2,-1,-1),-3}, {(-1,-1,-1),8}, {(0,-1,-1)\
,-6}, {(2,-1,-1),1};

Tx14 = {(-2,-2,-2),-1}, {(-1,-2,-2),2}, {(0,-2,-2),-1}, {(-2,-1,-2\
),2}, {(-1,-1,-2),-4}, {(0,-1,-2),2}, {(-2,0,-2),-1}, {(-1,0,-2),2\
}, {(0,0,-2),-1}, {(-2,-2,-1),1}, {(-1,-2,-1),-2}, {(0,-2,-1),1}, \
{(-2,-1,-1),-2}, {(-1,-1,-1),4}, {(0,-1,-1),-2}, {(-2,0,-1),1}, {(\
-1,0,-1),-2}, {(0,0,-1),1};

Tx15 = {(-2,-2,-2),-5}, {(-1,-2,-2),12}, {(0,-2,-2),-9}, {(1,-2,-2\
),2}, {(-2,-1,-2),8}, {(-1,-1,-2),-18}, {(0,-1,-2),12}, {(1,-1,-2)\
,-2}, {(-2,0,-2),-3}, {(-1,0,-2),6}, {(0,0,-2),-3}, {(-2,-2,-1),4}\
, {(-1,-2,-1),-9}, {(0,-2,-1),6}, {(1,-2,-1),-1}, {(-2,-1,-1),-6},\
 {(-1,-1,-1),12}, {(0,-1,-1),-6}, {(-2,0,-1),2}, {(-1,0,-1),-3}, {\
(1,0,-1),1};

Tx16 = {(-2,-2,-2),-15}, {(-1,-2,-2),40}, {(0,-2,-2),-36}, {(1,-2,\
-2),12}, {(2,-2,-2),-1}, {(-2,-1,-2),20}, {(-1,-1,-2),-48}, {(0,-1\
,-2),36}, {(1,-1,-2),-8}, {(-2,0,-2),-6}, {(-1,0,-2),12}, {(0,0,-2\
),-6}, {(-2,-2,-1),10}, {(-1,-2,-1),-24}, {(0,-2,-1),18}, {(1,-2,-\
1),-4}, {(-2,-1,-1),-12}, {(-1,-1,-1),24}, {(0,-1,-1),-12}, {(-2,0\
,-1),3}, {(-1,0,-1),-4}, {(2,0,-1),1};

Tx17 = {(-2,-2,-2),-1}, {(-1,-2,-2),1}, {(-2,-1,-2),3}, {(-1,-1,-2\
),-3}, {(-2,0,-2),-3}, {(-1,0,-2),3}, {(-2,1,-2),1}, {(-1,1,-2),-1\
}, {(-2,-2,-1),1}, {(-1,-2,-1),-1}, {(-2,-1,-1),-3}, {(-1,-1,-1),3\
}, {(-2,0,-1),3}, {(-1,0,-1),-3}, {(-2,1,-1),-1}, {(-1,1,-1),1};

Tx18 = {(-2,-2,-2),-5}, {(-1,-2,-2),8}, {(0,-2,-2),-3}, {(-2,-1,-2\
```

```
),12}, {(-1,-1,-2),-18}, {(0,-1,-2),6}, {(-2,0,-2),-9}, {(-1,0,-2)\
,12}, {(0,0,-2),-3}, {(-2,1,-2),2}, {(-1,1,-2),-2}, {(-2,-2,-1),4}\
, {(-1,-2,-1),-6}, {(0,-2,-1),2}, {(-2,-1,-1),-9}, {(-1,-1,-1),12}\
, {(0,-1,-1),-3}, {(-2,0,-1),6}, {(-1,0,-1),-6}, {(-2,1,-1),-1}, {\
(0,1,-1),1};

Tx19 = {(-2,-2,-2),-15}, {(-1,-2,-2),30}, {(0,-2,-2),-18}, {(1,-2,\
-2),3}, {(-2,-1,-2),30}, {(-1,-1,-2),-54}, {(0,-1,-2),27}, {(1,-1,\
-2),-3}, {(-2,0,-2),-18}, {(-1,0,-2),27}, {(0,0,-2),-9}, {(-2,1,-2\
),3}, {(-1,1,-2),-3}, {(-2,-2,-1),10}, {(-1,-2,-1),-18}, {(0,-2,-1\
),9}, {(1,-2,-1),-1}, {(-2,-1,-1),-18}, {(-1,-1,-1),27}, {(0,-1,-1\
),-9}, {(-2,0,-1),9}, {(-1,0,-1),-9}, {(-2,1,-1),-1}, {(1,1,-1),1};

Tx20 = {(-2,-2,-2),-35}, {(-1,-2,-2),80}, {(0,-2,-2),-60}, {(1,-2,\
-2),16}, {(2,-2,-2),-1}, {(-2,-1,-2),60}, {(-1,-1,-2),-120}, {(0,-\
1,-2),72}, {(1,-1,-2),-12}, {(-2,0,-2),-30}, {(-1,0,-2),48}, {(0,0\
,-2),-18}, {(-2,1,-2),4}, {(-1,1,-2),-4}, {(-2,-2,-1),20}, {(-1,-2\
,-1),-40}, {(0,-2,-1),24}, {(1,-2,-1),-4}, {(-2,-1,-1),-30}, {(-1,\
-1,-1),48}, {(0,-1,-1),-18}, {(-2,0,-1),12}, {(-1,0,-1),-12}, {(-2\
,1,-1),-1}, {(2,1,-1),1};

Tx21 = {(-2,-2,-2),-1}, {(-2,-1,-2),4}, {(-2,0,-2),-6}, {(-2,1,-2)\
,4}, {(-2,2,-2),-1}, {(-2,-2,-1),1}, {(-2,-1,-1),-4}, {(-2,0,-1),6\
}, {(-2,1,-1),-4}, {(-2,2,-1),1};

Tx22 = {(-2,-2,-2),-5}, {(-1,-2,-2),4}, {(-2,-1,-2),16}, {(-1,-1,-\
2),-12}, {(-2,0,-2),-18}, {(-1,0,-2),12}, {(-2,1,-2),8}, {(-1,1,-2\
),-4}, {(-2,2,-2),-1}, {(-2,-2,-1),4}, {(-1,-2,-1),-3}, {(-2,-1,-1\
),-12}, {(-1,-1,-1),8}, {(-2,0,-1),12}, {(-1,0,-1),-6}, {(-2,1,-1)\
,-4}, {(-1,2,-1),1};

Tx23 = {(-2,-2,-2),-15}, {(-1,-2,-2),20}, {(0,-2,-2),-6}, {(-2,-1,\
-2),40}, {(-1,-1,-2),-48}, {(0,-1,-2),12}, {(-2,0,-2),-36}, {(-1,0\
,-2),36}, {(0,0,-2),-6}, {(-2,1,-2),12}, {(-1,1,-2),-8}, {(-2,2,-2\
),-1}, {(-2,-2,-1),10}, {(-1,-2,-1),-12}, {(0,-2,-1),3}, {(-2,-1,-\
1),-24}, {(-1,-1,-1),24}, {(0,-1,-1),-4}, {(-2,0,-1),18}, {(-1,0,-\
1),-12}, {(-2,1,-1),-4}, {(0,2,-1),1};

Tx24 = {(-2,-2,-2),-35}, {(-1,-2,-2),60}, {(0,-2,-2),-30}, {(1,-2,\
-2),4}, {(-2,-1,-2),80}, {(-1,-1,-2),-120}, {(0,-1,-2),48}, {(1,-1\
```

```
,-2),-4}, {(-2,0,-2),-60}, {(-1,0,-2),72}, {(0,0,-2),-18}, {(-2,1,\
-2),16}, {(-1,1,-2),-12}, {(-2,2,-2),-1}, {(-2,-2,-1),20}, {(-1,-2\
,-1),-30}, {(0,-2,-1),12}, {(1,-2,-1),-1}, {(-2,-1,-1),-40}, {(-1,\
-1,-1),48}, {(0,-1,-1),-12}, {(-2,0,-1),24}, {(-1,0,-1),-18}, {(-2\
,1,-1),-4}, {(1,2,-1),1};
```

```
Tx25 = {(-2,-2,-2),-70}, {(-1,-2,-2),140}, {(0,-2,-2),-90}, {(1,-2\
,-2),20}, {(2,-2,-2),-1}, {(-2,-1,-2),140}, {(-1,-1,-2),-240}, {(0\
,-1,-2),120}, {(1,-1,-2),-16}, {(-2,0,-2),-90}, {(-1,0,-2),120}, {\
(0,0,-2),-36}, {(-2,1,-2),20}, {(-1,1,-2),-16}, {(-2,2,-2),-1}, {(\
-2,-2,-1),35}, {(-1,-2,-1),-60}, {(0,-2,-1),30}, {(1,-2,-1),-4}, {\
(-2,-1,-1),-60}, {(-1,-1,-1),80}, {(0,-1,-1),-24}, {(-2,0,-1),30},\
 {(-1,0,-1),-24}, {(-2,1,-1),-4}, {(2,2,-1),1};
```

```
Tx26 = {(-2,-2,-2),-1}, {(-1,-2,-2),3}, {(0,-2,-2),-3}, {(1,-2,-2)\
,1}, {(-2,-2,-1),2}, {(-1,-2,-1),-6}, {(0,-2,-1),6}, {(1,-2,-1),-2\
}, {(-2,-2,0),-1}, {(-1,-2,0),3}, {(0,-2,0),-3}, {(1,-2,0),1};
```

```
Tx27 = {(-2,-2,-2),-5}, {(-1,-2,-2),16}, {(0,-2,-2),-18}, {(1,-2,-\
2),8}, {(2,-2,-2),-1}, {(-2,-2,-1),8}, {(-1,-2,-1),-24}, {(0,-2,-1\
),24}, {(1,-2,-1),-8}, {(-2,-2,0),-3}, {(-1,-2,0),8}, {(0,-2,0),-6\
}, {(2,-2,0),1};
```

```
Tx28 = {(-2,-2,-2),-1}, {(-1,-2,-2),2}, {(0,-2,-2),-1}, {(-2,-1,-2\
),1}, {(-1,-1,-2),-2}, {(0,-1,-2),1}, {(-2,-2,-1),2}, {(-1,-2,-1),\
-4}, {(0,-2,-1),2}, {(-2,-1,-1),-2}, {(-1,-1,-1),4}, {(0,-1,-1),-2\
}, {(-2,-2,0),-1}, {(-1,-2,0),2}, {(0,-2,0),-1}, {(-2,-1,0),1}, {(\
-1,-1,0),-2}, {(0,-1,0),1};
```

```
Tx29 = {(-2,-2,-2),-5}, {(-1,-2,-2),12}, {(0,-2,-2),-9}, {(1,-2,-2\
),2}, {(-2,-1,-2),4}, {(-1,-1,-2),-9}, {(0,-1,-2),6}, {(1,-1,-2),-\
1}, {(-2,-2,-1),8}, {(-1,-2,-1),-18}, {(0,-2,-1),12}, {(1,-2,-1),-\
2}, {(-2,-1,-1),-6}, {(-1,-1,-1),12}, {(0,-1,-1),-6}, {(-2,-2,0),-\
3}, {(-1,-2,0),6}, {(0,-2,0),-3}, {(-2,-1,0),2}, {(-1,-1,0),-3}, {\
(1,-1,0),1};
```

```
Tx30 = {(-2,-2,-2),-15}, {(-1,-2,-2),40}, {(0,-2,-2),-36}, {(1,-2,\
-2),12}, {(2,-2,-2),-1}, {(-2,-1,-2),10}, {(-1,-1,-2),-24}, {(0,-1\
,-2),18}, {(1,-1,-2),-4}, {(-2,-2,-1),20}, {(-1,-2,-1),-48}, {(0,-\
2,-1),36}, {(1,-2,-1),-8}, {(-2,-1,-1),-12}, {(-1,-1,-1),24}, {(0,\
```

```
-1,-1),-12}, {(-2,-2,0),-6}, {(-1,-2,0),12}, {(0,-2,0),-6}, {(-2,-\
1,0),3}, {(-1,-1,0),-4}, {(2,-1,0),1};

Tx31 = {(-2,-2,-2),-1}, {(-1,-2,-2),1}, {(-2,-1,-2),2}, {(-1,-1,-2\
),-2}, {(-2,0,-2),-1}, {(-1,0,-2),1}, {(-2,-2,-1),2}, {(-1,-2,-1),\
-2}, {(-2,-1,-1),-4}, {(-1,-1,-1),4}, {(-2,0,-1),2}, {(-1,0,-1),-2\
}, {(-2,-2,0),-1}, {(-1,-2,0),1}, {(-2,-1,0),2}, {(-1,-1,0),-2}, {\
(-2,0,0),-1}, {(-1,0,0),1};

Tx32 = {(-2,-2,-2),-5}, {(-1,-2,-2),8}, {(0,-2,-2),-3}, {(-2,-1,-2\
),8}, {(-1,-1,-2),-12}, {(0,-1,-2),4}, {(-2,0,-2),-3}, {(-1,0,-2),\
4}, {(0,0,-2),-1}, {(-2,-2,-1),8}, {(-1,-2,-1),-12}, {(0,-2,-1),4}\
, {(-2,-1,-1),-12}, {(-1,-1,-1),16}, {(0,-1,-1),-4}, {(-2,0,-1),4}\
, {(-1,0,-1),-4}, {(-2,-2,0),-3}, {(-1,-2,0),4}, {(0,-2,0),-1}, {(\
-2,-1,0),4}, {(-1,-1,0),-4}, {(-2,0,0),-1}, {(0,0,0),1};

Tx33 = {(-2,-2,-2),-15}, {(-1,-2,-2),30}, {(0,-2,-2),-18}, {(1,-2,\
-2),3}, {(-2,-1,-2),20}, {(-1,-1,-2),-36}, {(0,-1,-2),18}, {(1,-1,\
-2),-2}, {(-2,0,-2),-6}, {(-1,0,-2),9}, {(0,0,-2),-3}, {(-2,-2,-1)\
,20}, {(-1,-2,-1),-36}, {(0,-2,-1),18}, {(1,-2,-1),-2}, {(-2,-1,-1\
),-24}, {(-1,-1,-1),36}, {(0,-1,-1),-12}, {(-2,0,-1),6}, {(-1,0,-1\
),-6}, {(-2,-2,0),-6}, {(-1,-2,0),9}, {(0,-2,0),-3}, {(-2,-1,0),6}\
, {(-1,-1,0),-6}, {(-2,0,0),-1}, {(1,0,0),1};

Tx34 = {(-2,-2,-2),-35}, {(-1,-2,-2),80}, {(0,-2,-2),-60}, {(1,-2,\
-2),16}, {(2,-2,-2),-1}, {(-2,-1,-2),40}, {(-1,-1,-2),-80}, {(0,-1\
,-2),48}, {(1,-1,-2),-8}, {(-2,0,-2),-10}, {(-1,0,-2),16}, {(0,0,-\
2),-6}, {(-2,-2,-1),40}, {(-1,-2,-1),-80}, {(0,-2,-1),48}, {(1,-2,\
-1),-8}, {(-2,-1,-1),-40}, {(-1,-1,-1),64}, {(0,-1,-1),-24}, {(-2,\
0,-1),8}, {(-1,0,-1),-8}, {(-2,-2,0),-10}, {(-1,-2,0),16}, {(0,-2,\
0),-6}, {(-2,-1,0),8}, {(-1,-1,0),-8}, {(-2,0,0),-1}, {(2,0,0),1};

Tx35 = {(-2,-2,-2),-1}, {(-2,-1,-2),3}, {(-2,0,-2),-3}, {(-2,1,-2)\
,1}, {(-2,-2,-1),2}, {(-2,-1,-1),-6}, {(-2,0,-1),6}, {(-2,1,-1),-2\
}, {(-2,-2,0),-1}, {(-2,-1,0),3}, {(-2,0,0),-3}, {(-2,1,0),1};

Tx36 = {(-2,-2,-2),-5}, {(-1,-2,-2),4}, {(-2,-1,-2),12}, {(-1,-1,-\
2),-9}, {(-2,0,-2),-9}, {(-1,0,-2),6}, {(-2,1,-2),2}, {(-1,1,-2),-\
1}, {(-2,-2,-1),8}, {(-1,-2,-1),-6}, {(-2,-1,-1),-18}, {(-1,-1,-1)\
,12}, {(-2,0,-1),12}, {(-1,0,-1),-6}, {(-2,1,-1),-2}, {(-2,-2,0),-\
```

```
3}, {(-1,-2,0),2}, {(-2,-1,0),6}, {(-1,-1,0),-3}, {(-2,0,0),-3}, {\
(-1,1,0),1};

Tx37 = {(-2,-2,-2),-15}, {(-1,-2,-2),20}, {(0,-2,-2),-6}, {(-2,-1,\
-2),30}, {(-1,-1,-2),-36}, {(0,-1,-2),9}, {(-2,0,-2),-18}, {(-1,0,\
-2),18}, {(0,0,-2),-3}, {(-2,1,-2),3}, {(-1,1,-2),-2}, {(-2,-2,-1)\
,20}, {(-1,-2,-1),-24}, {(0,-2,-1),6}, {(-2,-1,-1),-36}, {(-1,-1,-\
1),36}, {(0,-1,-1),-6}, {(-2,0,-1),18}, {(-1,0,-1),-12}, {(-2,1,-1\
),-2}, {(-2,-2,0),-6}, {(-1,-2,0),6}, {(0,-2,0),-1}, {(-2,-1,0),9}\
, {(-1,-1,0),-6}, {(-2,0,0),-3}, {(0,1,0),1};

Tx38 = {(-2,-2,-2),-35}, {(-1,-2,-2),60}, {(0,-2,-2),-30}, {(1,-2,\
-2),4}, {(-2,-1,-2),60}, {(-1,-1,-2),-90}, {(0,-1,-2),36}, {(1,-1,\
-2),-3}, {(-2,0,-2),-30}, {(-1,0,-2),36}, {(0,0,-2),-9}, {(-2,1,-2\
),4}, {(-1,1,-2),-3}, {(-2,-2,-1),40}, {(-1,-2,-1),-60}, {(0,-2,-1\
),24}, {(1,-2,-1),-2}, {(-2,-1,-1),-60}, {(-1,-1,-1),72}, {(0,-1,-\
1),-18}, {(-2,0,-1),24}, {(-1,0,-1),-18}, {(-2,1,-1),-2}, {(-2,-2,\
0),-10}, {(-1,-2,0),12}, {(0,-2,0),-3}, {(-2,-1,0),12}, {(-1,-1,0)\
,-9}, {(-2,0,0),-3}, {(1,1,0),1};

Tx39 = {(-2,-2,-2),-70}, {(-1,-2,-2),140}, {(0,-2,-2),-90}, {(1,-2\
,-2),20}, {(2,-2,-2),-1}, {(-2,-1,-2),105}, {(-1,-1,-2),-180}, {(0\
,-1,-2),90}, {(1,-1,-2),-12}, {(-2,0,-2),-45}, {(-1,0,-2),60}, {(0\
,0,-2),-18}, {(-2,1,-2),5}, {(-1,1,-2),-4}, {(-2,-2,-1),70}, {(-1,\
-2,-1),-120}, {(0,-2,-1),60}, {(1,-2,-1),-8}, {(-2,-1,-1),-90}, {(\
-1,-1,-1),120}, {(0,-1,-1),-36}, {(-2,0,-1),30}, {(-1,0,-1),-24}, \
{(-2,1,-1),-2}, {(-2,-2,0),-15}, {(-1,-2,0),20}, {(0,-2,0),-6}, {(\
-2,-1,0),15}, {(-1,-1,0),-12}, {(-2,0,0),-3}, {(2,1,0),1};

Tx40 = {(-2,-2,-2),-5}, {(-2,-1,-2),16}, {(-2,0,-2),-18}, {(-2,1,-\
2),8}, {(-2,2,-2),-1}, {(-2,-2,-1),8}, {(-2,-1,-1),-24}, {(-2,0,-1\
),24}, {(-2,1,-1),-8}, {(-2,-2,0),-3}, {(-2,-1,0),8}, {(-2,0,0),-6\
}, {(-2,2,0),1};

Tx41 = {(-2,-2,-2),-15}, {(-1,-2,-2),10}, {(-2,-1,-2),40}, {(-1,-1\
,-2),-24}, {(-2,0,-2),-36}, {(-1,0,-2),18}, {(-2,1,-2),12}, {(-1,1\
,-2),-4}, {(-2,2,-2),-1}, {(-2,-2,-1),20}, {(-1,-2,-1),-12}, {(-2,\
-1,-1),-48}, {(-1,-1,-1),24}, {(-2,0,-1),36}, {(-1,0,-1),-12}, {(-\
2,1,-1),-8}, {(-2,-2,0),-6}, {(-1,-2,0),3}, {(-2,-1,0),12}, {(-1,-\
1,0),-4}, {(-2,0,0),-6}, {(-1,2,0),1};
```

```
Tx42 = {(-2,-2,-2),-35}, {(-1,-2,-2),40}, {(0,-2,-2),-10}, {(-2,-1\
,-2),80}, {(-1,-1,-2),-80}, {(0,-1,-2),16}, {(-2,0,-2),-60}, {(-1,\
0,-2),48}, {(0,0,-2),-6}, {(-2,1,-2),16}, {(-1,1,-2),-8}, {(-2,2,-\
2),-1}, {(-2,-2,-1),40}, {(-1,-2,-1),-40}, {(0,-2,-1),8}, {(-2,-1,\
-1),-80}, {(-1,-1,-1),64}, {(0,-1,-1),-8}, {(-2,0,-1),48}, {(-1,0,\
-1),-24}, {(-2,1,-1),-8}, {(-2,-2,0),-10}, {(-1,-2,0),8}, {(0,-2,0\
),-1}, {(-2,-1,0),16}, {(-1,-1,0),-8}, {(-2,0,0),-6}, {(0,2,0),1};

Tx43 = {(-2,-2,-2),-70}, {(-1,-2,-2),105}, {(0,-2,-2),-45}, {(1,-2\
,-2),5}, {(-2,-1,-2),140}, {(-1,-1,-2),-180}, {(0,-1,-2),60}, {(1,\
-1,-2),-4}, {(-2,0,-2),-90}, {(-1,0,-2),90}, {(0,0,-2),-18}, {(-2,\
1,-2),20}, {(-1,1,-2),-12}, {(-2,2,-2),-1}, {(-2,-2,-1),70}, {(-1,\
-2,-1),-90}, {(0,-2,-1),30}, {(1,-2,-1),-2}, {(-2,-1,-1),-120}, {(\
-1,-1,-1),120}, {(0,-1,-1),-24}, {(-2,0,-1),60}, {(-1,0,-1),-36}, \
{(-2,1,-1),-8}, {(-2,-2,0),-15}, {(-1,-2,0),15}, {(0,-2,0),-3}, {(\
-2,-1,0),20}, {(-1,-1,0),-12}, {(-2,0,0),-6}, {(1,2,0),1};

Tx44 = {(-2,-2,-2),-126}, {(-1,-2,-2),224}, {(0,-2,-2),-126}, {(1,\
-2,-2),24}, {(2,-2,-2),-1}, {(-2,-1,-2),224}, {(-1,-1,-2),-336}, {\
(0,-1,-2),144}, {(1,-1,-2),-16}, {(-2,0,-2),-126}, {(-1,0,-2),144}\
, {(0,0,-2),-36}, {(-2,1,-2),24}, {(-1,1,-2),-16}, {(-2,2,-2),-1},\
 {(-2,-2,-1),112}, {(-1,-2,-1),-168}, {(0,-2,-1),72}, {(1,-2,-1),-\
8}, {(-2,-1,-1),-168}, {(-1,-1,-1),192}, {(0,-1,-1),-48}, {(-2,0,-\
1),72}, {(-1,0,-1),-48}, {(-2,1,-1),-8}, {(-2,-2,0),-21}, {(-1,-2,\
0),24}, {(0,-2,0),-6}, {(-2,-1,0),24}, {(-1,-1,0),-16}, {(-2,0,0),\
-6}, {(2,2,0),1};

Tx45 = {(-2,-2,-2),-1}, {(-1,-2,-2),2}, {(0,-2,-2),-1}, {(-2,-2,-1\
),3}, {(-1,-2,-1),-6}, {(0,-2,-1),3}, {(-2,-2,0),-3}, {(-1,-2,0),6\
}, {(0,-2,0),-3}, {(-2,-2,1),1}, {(-1,-2,1),-2}, {(0,-2,1),1};

Tx46 = {(-2,-2,-2),-5}, {(-1,-2,-2),12}, {(0,-2,-2),-9}, {(1,-2,-2\
),2}, {(-2,-2,-1),12}, {(-1,-2,-1),-27}, {(0,-2,-1),18}, {(1,-2,-1\
),-3}, {(-2,-2,0),-9}, {(-1,-2,0),18}, {(0,-2,0),-9}, {(-2,-2,1),2\
}, {(-1,-2,1),-3}, {(1,-2,1),1};

Tx47 = {(-2,-2,-2),-15}, {(-1,-2,-2),40}, {(0,-2,-2),-36}, {(1,-2,\
-2),12}, {(2,-2,-2),-1}, {(-2,-2,-1),30}, {(-1,-2,-1),-72}, {(0,-2\
,-1),54}, {(1,-2,-1),-12}, {(-2,-2,0),-18}, {(-1,-2,0),36}, {(0,-2\
```

```
,0),-18}, {(-2,-2,1),3}, {(-1,-2,1),-4}, {(2,-2,1),1};

Tx48 = {(-2,-2,-2),-1}, {(-1,-2,-2),1}, {(-2,-1,-2),1}, {(-1,-1,-2\
),-1}, {(-2,-2,-1),3}, {(-1,-2,-1),-3}, {(-2,-1,-1),-3}, {(-1,-1,-\
1),3}, {(-2,-2,0),-3}, {(-1,-2,0),3}, {(-2,-1,0),3}, {(-1,-1,0),-3\
}, {(-2,-2,1),1}, {(-1,-2,1),-1}, {(-2,-1,1),-1}, {(-1,-1,1),1};

Tx49 = {(-2,-2,-2),-5}, {(-1,-2,-2),8}, {(0,-2,-2),-3}, {(-2,-1,-2\
),4}, {(-1,-1,-2),-6}, {(0,-1,-2),2}, {(-2,-2,-1),12}, {(-1,-2,-1)\
,-18}, {(0,-2,-1),6}, {(-2,-1,-1),-9}, {(-1,-1,-1),12}, {(0,-1,-1)\
,-3}, {(-2,-2,0),-9}, {(-1,-2,0),12}, {(0,-2,0),-3}, {(-2,-1,0),6}\
, {(-1,-1,0),-6}, {(-2,-2,1),2}, {(-1,-2,1),-2}, {(-2,-1,1),-1}, {\
(0,-1,1),1};

Tx50 = {(-2,-2,-2),-15}, {(-1,-2,-2),30}, {(0,-2,-2),-18}, {(1,-2,\
-2),3}, {(-2,-1,-2),10}, {(-1,-1,-2),-18}, {(0,-1,-2),9}, {(1,-1,-\
2),-1}, {(-2,-2,-1),30}, {(-1,-2,-1),-54}, {(0,-2,-1),27}, {(1,-2,\
-1),-3}, {(-2,-1,-1),-18}, {(-1,-1,-1),27}, {(0,-1,-1),-9}, {(-2,-\
2,0),-18}, {(-1,-2,0),27}, {(0,-2,0),-9}, {(-2,-1,0),9}, {(-1,-1,0\
),-9}, {(-2,-2,1),3}, {(-1,-2,1),-3}, {(-2,-1,1),-1}, {(1,-1,1),1};

Tx51 = {(-2,-2,-2),-35}, {(-1,-2,-2),80}, {(0,-2,-2),-60}, {(1,-2,\
-2),16}, {(2,-2,-2),-1}, {(-2,-1,-2),20}, {(-1,-1,-2),-40}, {(0,-1\
,-2),24}, {(1,-1,-2),-4}, {(-2,-2,-1),60}, {(-1,-2,-1),-120}, {(0,\
-2,-1),72}, {(1,-2,-1),-12}, {(-2,-1,-1),-30}, {(-1,-1,-1),48}, {(\
0,-1,-1),-18}, {(-2,-2,0),-30}, {(-1,-2,0),48}, {(0,-2,0),-18}, {(\
-2,-1,0),12}, {(-1,-1,0),-12}, {(-2,-2,1),4}, {(-1,-2,1),-4}, {(-2\
,-1,1),-1}, {(2,-1,1),1};

Tx52 = {(-2,-2,-2),-1}, {(-2,-1,-2),2}, {(-2,0,-2),-1}, {(-2,-2,-1\
),3}, {(-2,-1,-1),-6}, {(-2,0,-1),3}, {(-2,-2,0),-3}, {(-2,-1,0),6\
}, {(-2,0,0),-3}, {(-2,-2,1),1}, {(-2,-1,1),-2}, {(-2,0,1),1};

Tx53 = {(-2,-2,-2),-5}, {(-1,-2,-2),4}, {(-2,-1,-2),8}, {(-1,-1,-2\
),-6}, {(-2,0,-2),-3}, {(-1,0,-2),2}, {(-2,-2,-1),12}, {(-1,-2,-1)\
,-9}, {(-2,-1,-1),-18}, {(-1,-1,-1),12}, {(-2,0,-1),6}, {(-1,0,-1)\
,-3}, {(-2,-2,0),-9}, {(-1,-2,0),6}, {(-2,-1,0),12}, {(-1,-1,0),-6\
}, {(-2,0,0),-3}, {(-2,-2,1),2}, {(-1,-2,1),-1}, {(-2,-1,1),-2}, {\
(-1,0,1),1};
```

```
Tx54 = {(-2,-2,-2),-15}, {(-1,-2,-2),20}, {(0,-2,-2),-6}, {(-2,-1,\
-2),20}, {(-1,-1,-2),-24}, {(0,-1,-2),6}, {(-2,0,-2),-6}, {(-1,0,-\
2),6}, {(0,0,-2),-1}, {(-2,-2,-1),30}, {(-1,-2,-1),-36}, {(0,-2,-1\
),9}, {(-2,-1,-1),-36}, {(-1,-1,-1),36}, {(0,-1,-1),-6}, {(-2,0,-1\
),9}, {(-1,0,-1),-6}, {(-2,-2,0),-18}, {(-1,-2,0),18}, {(0,-2,0),-\
3}, {(-2,-1,0),18}, {(-1,-1,0),-12}, {(-2,0,0),-3}, {(-2,-2,1),3},\
 {(-1,-2,1),-2}, {(-2,-1,1),-2}, {(0,0,1),1};

Tx55 = {(-2,-2,-2),-35}, {(-1,-2,-2),60}, {(0,-2,-2),-30}, {(1,-2,\
-2),4}, {(-2,-1,-2),40}, {(-1,-1,-2),-60}, {(0,-1,-2),24}, {(1,-1,\
-2),-2}, {(-2,0,-2),-10}, {(-1,0,-2),12}, {(0,0,-2),-3}, {(-2,-2,-\
1),60}, {(-1,-2,-1),-90}, {(0,-2,-1),36}, {(1,-2,-1),-3}, {(-2,-1,\
-1),-60}, {(-1,-1,-1),72}, {(0,-1,-1),-18}, {(-2,0,-1),12}, {(-1,0\
,-1),-9}, {(-2,-2,0),-30}, {(-1,-2,0),36}, {(0,-2,0),-9}, {(-2,-1,\
0),24}, {(-1,-1,0),-18}, {(-2,0,0),-3}, {(-2,-2,1),4}, {(-1,-2,1),\
-3}, {(-2,-1,1),-2}, {(1,0,1),1};

Tx56 = {(-2,-2,-2),-70}, {(-1,-2,-2),140}, {(0,-2,-2),-90}, {(1,-2\
,-2),20}, {(2,-2,-2),-1}, {(-2,-1,-2),70}, {(-1,-1,-2),-120}, {(0,\
-1,-2),60}, {(1,-1,-2),-8}, {(-2,0,-2),-15}, {(-1,0,-2),20}, {(0,0\
,-2),-6}, {(-2,-2,-1),105}, {(-1,-2,-1),-180}, {(0,-2,-1),90}, {(1\
,-2,-1),-12}, {(-2,-1,-1),-90}, {(-1,-1,-1),120}, {(0,-1,-1),-36},\
 {(-2,0,-1),15}, {(-1,0,-1),-12}, {(-2,-2,0),-45}, {(-1,-2,0),60},\
 {(0,-2,0),-18}, {(-2,-1,0),30}, {(-1,-1,0),-24}, {(-2,0,0),-3}, {\
(-2,-2,1),5}, {(-1,-2,1),-4}, {(-2,-1,1),-2}, {(2,0,1),1};

Tx57 = {(-2,-2,-2),-5}, {(-2,-1,-2),12}, {(-2,0,-2),-9}, {(-2,1,-2\
),2}, {(-2,-2,-1),12}, {(-2,-1,-1),-27}, {(-2,0,-1),18}, {(-2,1,-1\
),-3}, {(-2,-2,0),-9}, {(-2,-1,0),18}, {(-2,0,0),-9}, {(-2,-2,1),2\
}, {(-2,-1,1),-3}, {(-2,1,1),1};

Tx58 = {(-2,-2,-2),-15}, {(-1,-2,-2),10}, {(-2,-1,-2),30}, {(-1,-1\
,-2),-18}, {(-2,0,-2),-18}, {(-1,0,-2),9}, {(-2,1,-2),3}, {(-1,1,-\
2),-1}, {(-2,-2,-1),30}, {(-1,-2,-1),-18}, {(-2,-1,-1),-54}, {(-1,\
-1,-1),27}, {(-2,0,-1),27}, {(-1,0,-1),-9}, {(-2,1,-1),-3}, {(-2,-\
2,0),-18}, {(-1,-2,0),9}, {(-2,-1,0),27}, {(-1,-1,0),-9}, {(-2,0,0\
),-9}, {(-2,-2,1),3}, {(-1,-2,1),-1}, {(-2,-1,1),-3}, {(-1,1,1),1};

Tx59 = {(-2,-2,-2),-35}, {(-1,-2,-2),40}, {(0,-2,-2),-10}, {(-2,-1\
,-2),60}, {(-1,-1,-2),-60}, {(0,-1,-2),12}, {(-2,0,-2),-30}, {(-1,\
```

```
0,−2),24}, {(0,0,−2),−3}, {(−2,1,−2),4}, {(−1,1,−2),−2}, {(−2,−2,−\
1),60}, {(−1,−2,−1),−60}, {(0,−2,−1),12}, {(−2,−1,−1),−90}, {(−1,−\
1,−1),72}, {(0,−1,−1),−9}, {(−2,0,−1),36}, {(−1,0,−1),−18}, {(−2,1\
,−1),−3}, {(−2,−2,0),−30}, {(−1,−2,0),24}, {(0,−2,0),−3}, {(−2,−1,\
0),36}, {(−1,−1,0),−18}, {(−2,0,0),−9}, {(−2,−2,1),4}, {(−1,−2,1),\
−2}, {(−2,−1,1),−3}, {(0,1,1),1};

Tx60 = {(−2,−2,−2),−70}, {(−1,−2,−2),105}, {(0,−2,−2),−45}, {(1,−2\
,−2),5}, {(−2,−1,−2),105}, {(−1,−1,−2),−135}, {(0,−1,−2),45}, {(1,\
−1,−2),−3}, {(−2,0,−2),−45}, {(−1,0,−2),45}, {(0,0,−2),−9}, {(−2,1\
,−2),5}, {(−1,1,−2),−3}, {(−2,−2,−1),105}, {(−1,−2,−1),−135}, {(0,\
−2,−1),45}, {(1,−2,−1),−3}, {(−2,−1,−1),−135}, {(−1,−1,−1),135}, {\
(0,−1,−1),−27}, {(−2,0,−1),45}, {(−1,0,−1),−27}, {(−2,1,−1),−3}, {\
(−2,−2,0),−45}, {(−1,−2,0),45}, {(0,−2,0),−9}, {(−2,−1,0),45}, {(−\
1,−1,0),−27}, {(−2,0,0),−9}, {(−2,−2,1),5}, {(−1,−2,1),−3}, {(−2,−\
1,1),−3}, {(1,1,1),1};

Tx61 = {(−2,−2,−2),−126}, {(−1,−2,−2),224}, {(0,−2,−2),−126}, {(1,\
−2,−2),24}, {(2,−2,−2),−1}, {(−2,−1,−2),168}, {(−1,−1,−2),−252}, {\
(0,−1,−2),108}, {(1,−1,−2),−12}, {(−2,0,−2),−63}, {(−1,0,−2),72}, \
{(0,0,−2),−18}, {(−2,1,−2),6}, {(−1,1,−2),−4}, {(−2,−2,−1),168}, {\
(−1,−2,−1),−252}, {(0,−2,−1),108}, {(1,−2,−1),−12}, {(−2,−1,−1),−1\
89}, {(−1,−1,−1),216}, {(0,−1,−1),−54}, {(−2,0,−1),54}, {(−1,0,−1)\
,−36}, {(−2,1,−1),−3}, {(−2,−2,0),−63}, {(−1,−2,0),72}, {(0,−2,0),\
−18}, {(−2,−1,0),54}, {(−1,−1,0),−36}, {(−2,0,0),−9}, {(−2,−2,1),6\
}, {(−1,−2,1),−4}, {(−2,−1,1),−3}, {(2,1,1),1};

Tx62 = {(−2,−2,−2),−15}, {(−2,−1,−2),40}, {(−2,0,−2),−36}, {(−2,1,\
−2),12}, {(−2,2,−2),−1}, {(−2,−2,−1),30}, {(−2,−1,−1),−72}, {(−2,0\
,−1),54}, {(−2,1,−1),−12}, {(−2,−2,0),−18}, {(−2,−1,0),36}, {(−2,0\
,0),−18}, {(−2,−2,1),3}, {(−2,−1,1),−4}, {(−2,2,1),1};

Tx63 = {(−2,−2,−2),−35}, {(−1,−2,−2),20}, {(−2,−1,−2),80}, {(−1,−1\
,−2),−40}, {(−2,0,−2),−60}, {(−1,0,−2),24}, {(−2,1,−2),16}, {(−1,1\
,−2),−4}, {(−2,2,−2),−1}, {(−2,−2,−1),60}, {(−1,−2,−1),−30}, {(−2,\
−1,−1),−120}, {(−1,−1,−1),48}, {(−2,0,−1),72}, {(−1,0,−1),−18}, {(\
−2,1,−1),−12}, {(−2,−2,0),−30}, {(−1,−2,0),12}, {(−2,−1,0),48}, {(\
−1,−1,0),−12}, {(−2,0,0),−18}, {(−2,−2,1),4}, {(−1,−2,1),−1}, {(−2\
,−1,1),−4}, {(−1,2,1),1};
```

```
Tx64 = {(-2,-2,-2),-70}, {(-1,-2,-2),70}, {(0,-2,-2),-15}, {(-2,-1\
,-2),140}, {(-1,-1,-2),-120}, {(0,-1,-2),20}, {(-2,0,-2),-90}, {(-\
1,0,-2),60}, {(0,0,-2),-6}, {(-2,1,-2),20}, {(-1,1,-2),-8}, {(-2,2\
,-2),-1}, {(-2,-2,-1),105}, {(-1,-2,-1),-90}, {(0,-2,-1),15}, {(-2\
,-1,-1),-180}, {(-1,-1,-1),120}, {(0,-1,-1),-12}, {(-2,0,-1),90}, \
{(-1,0,-1),-36}, {(-2,1,-1),-12}, {(-2,-2,0),-45}, {(-1,-2,0),30},\
 {(0,-2,0),-3}, {(-2,-1,0),60}, {(-1,-1,0),-24}, {(-2,0,0),-18}, {\
(-2,-2,1),5}, {(-1,-2,1),-2}, {(-2,-1,1),-4}, {(0,2,1),1};

Tx65 = {(-2,-2,-2),-126}, {(-1,-2,-2),168}, {(0,-2,-2),-63}, {(1,-\
2,-2),6}, {(-2,-1,-2),224}, {(-1,-1,-2),-252}, {(0,-1,-2),72}, {(1\
,-1,-2),-4}, {(-2,0,-2),-126}, {(-1,0,-2),108}, {(0,0,-2),-18}, {(\
-2,1,-2),24}, {(-1,1,-2),-12}, {(-2,2,-2),-1}, {(-2,-2,-1),168}, {\
(-1,-2,-1),-189}, {(0,-2,-1),54}, {(1,-2,-1),-3}, {(-2,-1,-1),-252\
}, {(-1,-1,-1),216}, {(0,-1,-1),-36}, {(-2,0,-1),108}, {(-1,0,-1),\
-54}, {(-2,1,-1),-12}, {(-2,-2,0),-63}, {(-1,-2,0),54}, {(0,-2,0),\
-9}, {(-2,-1,0),72}, {(-1,-1,0),-36}, {(-2,0,0),-18}, {(-2,-2,1),6\
}, {(-1,-2,1),-3}, {(-2,-1,1),-4}, {(1,2,1),1};

Tx66 = {(-2,-2,-2),-210}, {(-1,-2,-2),336}, {(0,-2,-2),-168}, {(1,\
-2,-2),28}, {(2,-2,-2),-1}, {(-2,-1,-2),336}, {(-1,-1,-2),-448}, {\
(0,-1,-2),168}, {(1,-1,-2),-16}, {(-2,0,-2),-168}, {(-1,0,-2),168}\
, {(0,0,-2),-36}, {(-2,1,-2),28}, {(-1,1,-2),-16}, {(-2,2,-2),-1},\
 {(-2,-2,-1),252}, {(-1,-2,-1),-336}, {(0,-2,-1),126}, {(1,-2,-1),\
-12}, {(-2,-1,-1),-336}, {(-1,-1,-1),336}, {(0,-1,-1),-72}, {(-2,0\
,-1),126}, {(-1,0,-1),-72}, {(-2,1,-1),-12}, {(-2,-2,0),-84}, {(-1\
,-2,0),84}, {(0,-2,0),-18}, {(-2,-1,0),84}, {(-1,-1,0),-48}, {(-2,\
0,0),-18}, {(-2,-2,1),7}, {(-1,-2,1),-4}, {(-2,-1,1),-4}, {(2,2,1)\
,1};

Tx67 = {(-2,-2,-2),-1}, {(-1,-2,-2),1}, {(-2,-2,-1),4}, {(-1,-2,-1\
),-4}, {(-2,-2,0),-6}, {(-1,-2,0),6}, {(-2,-2,1),4}, {(-1,-2,1),-4\
}, {(-2,-2,2),-1}, {(-1,-2,2),1};

Tx68 = {(-2,-2,-2),-5}, {(-1,-2,-2),8}, {(0,-2,-2),-3}, {(-2,-2,-1\
),16}, {(-1,-2,-1),-24}, {(0,-2,-1),8}, {(-2,-2,0),-18}, {(-1,-2,0\
),24}, {(0,-2,0),-6}, {(-2,-2,1),8}, {(-1,-2,1),-8}, {(-2,-2,2),-1\
}, {(0,-2,2),1};

Tx69 = {(-2,-2,-2),-15}, {(-1,-2,-2),30}, {(0,-2,-2),-18}, {(1,-2,\
```

```
-2),3}, {(-2,-2,-1),40}, {(-1,-2,-1),-72}, {(0,-2,-1),36}, {(1,-2,\
-1),-4}, {(-2,-2,0),-36}, {(-1,-2,0),54}, {(0,-2,0),-18}, {(-2,-2,\
1),12}, {(-1,-2,1),-12}, {(-2,-2,2),-1}, {(1,-2,2),1};

Tx70 = {(-2,-2,-2),-35}, {(-1,-2,-2),80}, {(0,-2,-2),-60}, {(1,-2,\
-2),16}, {(2,-2,-2),-1}, {(-2,-2,-1),80}, {(-1,-2,-1),-160}, {(0,-\
2,-1),96}, {(1,-2,-1),-16}, {(-2,-2,0),-60}, {(-1,-2,0),96}, {(0,-\
2,0),-36}, {(-2,-2,1),16}, {(-1,-2,1),-16}, {(-2,-2,2),-1}, {(2,-2\
,2),1};

Tx71 = {(-2,-2,-2),-1}, {(-2,-1,-2),1}, {(-2,-2,-1),4}, {(-2,-1,-1\
),-4}, {(-2,-2,0),-6}, {(-2,-1,0),6}, {(-2,-2,1),4}, {(-2,-1,1),-4\
}, {(-2,-2,2),-1}, {(-2,-1,2),1};

Tx72 = {(-2,-2,-2),-5}, {(-1,-2,-2),4}, {(-2,-1,-2),4}, {(-1,-1,-2\
),-3}, {(-2,-2,-1),16}, {(-1,-2,-1),-12}, {(-2,-1,-1),-12}, {(-1,-\
1,-1),8}, {(-2,-2,0),-18}, {(-1,-2,0),12}, {(-2,-1,0),12}, {(-1,-1\
,0),-6}, {(-2,-2,1),8}, {(-1,-2,1),-4}, {(-2,-1,1),-4}, {(-2,-2,2)\
,-1}, {(-1,-1,2),1};

Tx73 = {(-2,-2,-2),-15}, {(-1,-2,-2),20}, {(0,-2,-2),-6}, {(-2,-1,\
-2),10}, {(-1,-1,-2),-12}, {(0,-1,-2),3}, {(-2,-2,-1),40}, {(-1,-2\
,-1),-48}, {(0,-2,-1),12}, {(-2,-1,-1),-24}, {(-1,-1,-1),24}, {(0,\
-1,-1),-4}, {(-2,-2,0),-36}, {(-1,-2,0),36}, {(0,-2,0),-6}, {(-2,-\
1,0),18}, {(-1,-1,0),-12}, {(-2,-2,1),12}, {(-1,-2,1),-8}, {(-2,-1\
,1),-4}, {(-2,-2,2),-1}, {(0,-1,2),1};

Tx74 = {(-2,-2,-2),-35}, {(-1,-2,-2),60}, {(0,-2,-2),-30}, {(1,-2,\
-2),4}, {(-2,-1,-2),20}, {(-1,-1,-2),-30}, {(0,-1,-2),12}, {(1,-1,\
-2),-1}, {(-2,-2,-1),80}, {(-1,-2,-1),-120}, {(0,-2,-1),48}, {(1,-\
2,-1),-4}, {(-2,-1,-1),-40}, {(-1,-1,-1),48}, {(0,-1,-1),-12}, {(-\
2,-2,0),-60}, {(-1,-2,0),72}, {(0,-2,0),-18}, {(-2,-1,0),24}, {(-1\
,-1,0),-18}, {(-2,-2,1),16}, {(-1,-2,1),-12}, {(-2,-1,1),-4}, {(-2\
,-2,2),-1}, {(1,-1,2),1};

Tx75 = {(-2,-2,-2),-70}, {(-1,-2,-2),140}, {(0,-2,-2),-90}, {(1,-2\
,-2),20}, {(2,-2,-2),-1}, {(-2,-1,-2),35}, {(-1,-1,-2),-60}, {(0,-\
1,-2),30}, {(1,-1,-2),-4}, {(-2,-2,-1),140}, {(-1,-2,-1),-240}, {(\
0,-2,-1),120}, {(1,-2,-1),-16}, {(-2,-1,-1),-60}, {(-1,-1,-1),80},\
 {(0,-1,-1),-24}, {(-2,-2,0),-90}, {(-1,-2,0),120}, {(0,-2,0),-36}\
```

```
, {(-2,-1,0),30}, {(-1,-1,0),-24}, {(-2,-2,1),20}, {(-1,-2,1),-16}\
, {(-2,-1,1),-4}, {(-2,-2,2),-1}, {(2,-1,2),1};

Tx76 = {(-2,-2,-2),-5}, {(-2,-1,-2),8}, {(-2,0,-2),-3}, {(-2,-2,-1\
),16}, {(-2,-1,-1),-24}, {(-2,0,-1),8}, {(-2,-2,0),-18}, {(-2,-1,0\
),24}, {(-2,0,0),-6}, {(-2,-2,1),8}, {(-2,-1,1),-8}, {(-2,-2,2),-1\
}, {(-2,0,2),1};

Tx77 = {(-2,-2,-2),-15}, {(-1,-2,-2),10}, {(-2,-1,-2),20}, {(-1,-1\
,-2),-12}, {(-2,0,-2),-6}, {(-1,0,-2),3}, {(-2,-2,-1),40}, {(-1,-2\
,-1),-24}, {(-2,-1,-1),-48}, {(-1,-1,-1),24}, {(-2,0,-1),12}, {(-1\
,0,-1),-4}, {(-2,-2,0),-36}, {(-1,-2,0),18}, {(-2,-1,0),36}, {(-1,\
-1,0),-12}, {(-2,0,0),-6}, {(-2,-2,1),12}, {(-1,-2,1),-4}, {(-2,-1\
,1),-8}, {(-2,-2,2),-1}, {(-1,0,2),1};

Tx78 = {(-2,-2,-2),-35}, {(-1,-2,-2),40}, {(0,-2,-2),-10}, {(-2,-1\
,-2),40}, {(-1,-1,-2),-40}, {(0,-1,-2),8}, {(-2,0,-2),-10}, {(-1,0\
,-2),8}, {(0,0,-2),-1}, {(-2,-2,-1),80}, {(-1,-2,-1),-80}, {(0,-2,\
-1),16}, {(-2,-1,-1),-80}, {(-1,-1,-1),64}, {(0,-1,-1),-8}, {(-2,0\
,-1),16}, {(-1,0,-1),-8}, {(-2,-2,0),-60}, {(-1,-2,0),48}, {(0,-2,\
0),-6}, {(-2,-1,0),48}, {(-1,-1,0),-24}, {(-2,0,0),-6}, {(-2,-2,1)\
,16}, {(-1,-2,1),-8}, {(-2,-1,1),-8}, {(-2,-2,2),-1}, {(0,0,2),1};

Tx79 = {(-2,-2,-2),-70}, {(-1,-2,-2),105}, {(0,-2,-2),-45}, {(1,-2\
,-2),5}, {(-2,-1,-2),70}, {(-1,-1,-2),-90}, {(0,-1,-2),30}, {(1,-1\
,-2),-2}, {(-2,0,-2),-15}, {(-1,0,-2),15}, {(0,0,-2),-3}, {(-2,-2,\
-1),140}, {(-1,-2,-1),-180}, {(0,-2,-1),60}, {(1,-2,-1),-4}, {(-2,\
-1,-1),-120}, {(-1,-1,-1),120}, {(0,-1,-1),-24}, {(-2,0,-1),20}, {\
(-1,0,-1),-12}, {(-2,-2,0),-90}, {(-1,-2,0),90}, {(0,-2,0),-18}, {\
(-2,-1,0),60}, {(-1,-1,0),-36}, {(-2,0,0),-6}, {(-2,-2,1),20}, {(-\
1,-2,1),-12}, {(-2,-1,1),-8}, {(-2,-2,2),-1}, {(1,0,2),1};

Tx80 = {(-2,-2,-2),-126}, {(-1,-2,-2),224}, {(0,-2,-2),-126}, {(1,\
-2,-2),24}, {(2,-2,-2),-1}, {(-2,-1,-2),112}, {(-1,-1,-2),-168}, {\
(0,-1,-2),72}, {(1,-1,-2),-8}, {(-2,0,-2),-21}, {(-1,0,-2),24}, {(\
0,0,-2),-6}, {(-2,-2,-1),224}, {(-1,-2,-1),-336}, {(0,-2,-1),144},\
 {(1,-2,-1),-16}, {(-2,-1,-1),-168}, {(-1,-1,-1),192}, {(0,-1,-1),\
-48}, {(-2,0,-1),24}, {(-1,0,-1),-16}, {(-2,-2,0),-126}, {(-1,-2,0\
),144}, {(0,-2,0),-36}, {(-2,-1,0),72}, {(-1,-1,0),-48}, {(-2,0,0)\
,-6}, {(-2,-2,1),24}, {(-1,-2,1),-16}, {(-2,-1,1),-8}, {(-2,-2,2),\
```

```
-1}, {(2,0,2),1};

Tx81 = {(-2,-2,-2),-15}, {(-2,-1,-2),30}, {(-2,0,-2),-18}, {(-2,1,\
-2),3}, {(-2,-2,-1),40}, {(-2,-1,-1),-72}, {(-2,0,-1),36}, {(-2,1,\
-1),-4}, {(-2,-2,0),-36}, {(-2,-1,0),54}, {(-2,0,0),-18}, {(-2,-2,\
1),12}, {(-2,-1,1),-12}, {(-2,-2,2),-1}, {(-2,1,2),1};

Tx82 = {(-2,-2,-2),-35}, {(-1,-2,-2),20}, {(-2,-1,-2),60}, {(-1,-1\
,-2),-30}, {(-2,0,-2),-30}, {(-1,0,-2),12}, {(-2,1,-2),4}, {(-1,1,\
-2),-1}, {(-2,-2,-1),80}, {(-1,-2,-1),-40}, {(-2,-1,-1),-120}, {(-\
1,-1,-1),48}, {(-2,0,-1),48}, {(-1,0,-1),-12}, {(-2,1,-1),-4}, {(-\
2,-2,0),-60}, {(-1,-2,0),24}, {(-2,-1,0),72}, {(-1,-1,0),-18}, {(-\
2,0,0),-18}, {(-2,-2,1),16}, {(-1,-2,1),-4}, {(-2,-1,1),-12}, {(-2\
,-2,2),-1}, {(-1,1,2),1};

Tx83 = {(-2,-2,-2),-70}, {(-1,-2,-2),70}, {(0,-2,-2),-15}, {(-2,-1\
,-2),105}, {(-1,-1,-2),-90}, {(0,-1,-2),15}, {(-2,0,-2),-45}, {(-1\
,0,-2),30}, {(0,0,-2),-3}, {(-2,1,-2),5}, {(-1,1,-2),-2}, {(-2,-2,\
-1),140}, {(-1,-2,-1),-120}, {(0,-2,-1),20}, {(-2,-1,-1),-180}, {(\
-1,-1,-1),120}, {(0,-1,-1),-12}, {(-2,0,-1),60}, {(-1,0,-1),-24}, \
{(-2,1,-1),-4}, {(-2,-2,0),-90}, {(-1,-2,0),60}, {(0,-2,0),-6}, {(\
-2,-1,0),90}, {(-1,-1,0),-36}, {(-2,0,0),-18}, {(-2,-2,1),20}, {(-\
1,-2,1),-8}, {(-2,-1,1),-12}, {(-2,-2,2),-1}, {(0,1,2),1};

Tx84 = {(-2,-2,-2),-126}, {(-1,-2,-2),168}, {(0,-2,-2),-63}, {(1,-\
2,-2),6}, {(-2,-1,-2),168}, {(-1,-1,-2),-189}, {(0,-1,-2),54}, {(1\
,-1,-2),-3}, {(-2,0,-2),-63}, {(-1,0,-2),54}, {(0,0,-2),-9}, {(-2,\
1,-2),6}, {(-1,1,-2),-3}, {(-2,-2,-1),224}, {(-1,-2,-1),-252}, {(0\
,-2,-1),72}, {(1,-2,-1),-4}, {(-2,-1,-1),-252}, {(-1,-1,-1),216}, \
{(0,-1,-1),-36}, {(-2,0,-1),72}, {(-1,0,-1),-36}, {(-2,1,-1),-4}, \
{(-2,-2,0),-126}, {(-1,-2,0),108}, {(0,-2,0),-18}, {(-2,-1,0),108}\
, {(-1,-1,0),-54}, {(-2,0,0),-18}, {(-2,-2,1),24}, {(-1,-2,1),-12}\
, {(-2,-1,1),-12}, {(-2,-2,2),-1}, {(1,1,2),1};

Tx85 = {(-2,-2,-2),-210}, {(-1,-2,-2),336}, {(0,-2,-2),-168}, {(1,\
-2,-2),28}, {(2,-2,-2),-1}, {(-2,-1,-2),252}, {(-1,-1,-2),-336}, {\
(0,-1,-2),126}, {(1,-1,-2),-12}, {(-2,0,-2),-84}, {(-1,0,-2),84}, \
{(0,0,-2),-18}, {(-2,1,-2),7}, {(-1,1,-2),-4}, {(-2,-2,-1),336}, {\
(-1,-2,-1),-448}, {(0,-2,-1),168}, {(1,-2,-1),-16}, {(-2,-1,-1),-3\
36}, {(-1,-1,-1),336}, {(0,-1,-1),-72}, {(-2,0,-1),84}, {(-1,0,-1)\
```

```
,−48}, {(−2,1,−1),−4}, {(−2,−2,0),−168}, {(−1,−2,0),168}, {(0,−2,0\
),−36}, {(−2,−1,0),126}, {(−1,−1,0),−72}, {(−2,0,0),−18}, {(−2,−2,\
1),28}, {(−1,−2,1),−16}, {(−2,−1,1),−12}, {(−2,−2,2),−1}, {(2,1,2)\
,1};

Tx86 = {(−2,−2,−2),−35}, {(−2,−1,−2),80}, {(−2,0,−2),−60}, {(−2,1,\
−2),16}, {(−2,2,−2),−1}, {(−2,−2,−1),80}, {(−2,−1,−1),−160}, {(−2,\
0,−1),96}, {(−2,1,−1),−16}, {(−2,−2,0),−60}, {(−2,−1,0),96}, {(−2,\
0,0),−36}, {(−2,−2,1),16}, {(−2,−1,1),−16}, {(−2,−2,2),−1}, {(−2,2\
,2),1};

Tx87 = {(−2,−2,−2),−70}, {(−1,−2,−2),35}, {(−2,−1,−2),140}, {(−1,−\
1,−2),−60}, {(−2,0,−2),−90}, {(−1,0,−2),30}, {(−2,1,−2),20}, {(−1,\
1,−2),−4}, {(−2,2,−2),−1}, {(−2,−2,−1),140}, {(−1,−2,−1),−60}, {(−\
2,−1,−1),−240}, {(−1,−1,−1),80}, {(−2,0,−1),120}, {(−1,0,−1),−24},\
 {(−2,1,−1),−16}, {(−2,−2,0),−90}, {(−1,−2,0),30}, {(−2,−1,0),120}\
, {(−1,−1,0),−24}, {(−2,0,0),−36}, {(−2,−2,1),20}, {(−1,−2,1),−4},\
 {(−2,−1,1),−16}, {(−2,−2,2),−1}, {(−1,2,2),1};

Tx88 = {(−2,−2,−2),−126}, {(−1,−2,−2),112}, {(0,−2,−2),−21}, {(−2,\
−1,−2),224}, {(−1,−1,−2),−168}, {(0,−1,−2),24}, {(−2,0,−2),−126}, \
{(−1,0,−2),72}, {(0,0,−2),−6}, {(−2,1,−2),24}, {(−1,1,−2),−8}, {(−\
2,2,−2),−1}, {(−2,−2,−1),224}, {(−1,−2,−1),−168}, {(0,−2,−1),24}, \
{(−2,−1,−1),−336}, {(−1,−1,−1),192}, {(0,−1,−1),−16}, {(−2,0,−1),1\
44}, {(−1,0,−1),−48}, {(−2,1,−1),−16}, {(−2,−2,0),−126}, {(−1,−2,0\
),72}, {(0,−2,0),−6}, {(−2,−1,0),144}, {(−1,−1,0),−48}, {(−2,0,0),\
−36}, {(−2,−2,1),24}, {(−1,−2,1),−8}, {(−2,−1,1),−16}, {(−2,−2,2),\
−1}, {(0,2,2),1};

Tx89 = {(−2,−2,−2),−210}, {(−1,−2,−2),252}, {(0,−2,−2),−84}, {(1,−\
2,−2),7}, {(−2,−1,−2),336}, {(−1,−1,−2),−336}, {(0,−1,−2),84}, {(1\
,−1,−2),−4}, {(−2,0,−2),−168}, {(−1,0,−2),126}, {(0,0,−2),−18}, {(\
−2,1,−2),28}, {(−1,1,−2),−12}, {(−2,2,−2),−1}, {(−2,−2,−1),336}, {\
(−1,−2,−1),−336}, {(0,−2,−1),84}, {(1,−2,−1),−4}, {(−2,−1,−1),−448\
}, {(−1,−1,−1),336}, {(0,−1,−1),−48}, {(−2,0,−1),168}, {(−1,0,−1),\
−72}, {(−2,1,−1),−16}, {(−2,−2,0),−168}, {(−1,−2,0),126}, {(0,−2,0\
),−18}, {(−2,−1,0),168}, {(−1,−1,0),−72}, {(−2,0,0),−36}, {(−2,−2,\
1),28}, {(−1,−2,1),−12}, {(−2,−1,1),−16}, {(−2,−2,2),−1}, {(1,2,2)\
,1};
```

```
Tx90 = {(-2,-2,-2),-330}, {(-1,-2,-2),480}, {(0,-2,-2),-216}, {(1,\
-2,-2),32}, {(2,-2,-2),-1}, {(-2,-1,-2),480}, {(-1,-1,-2),-576}, {\
(0,-1,-2),192}, {(1,-1,-2),-16}, {(-2,0,-2),-216}, {(-1,0,-2),192}\
, {(0,0,-2),-36}, {(-2,1,-2),32}, {(-1,1,-2),-16}, {(-2,2,-2),-1},\
 {(-2,-2,-1),480}, {(-1,-2,-1),-576}, {(0,-2,-1),192}, {(1,-2,-1),\
-16}, {(-2,-1,-1),-576}, {(-1,-1,-1),512}, {(0,-1,-1),-96}, {(-2,0\
,-1),192}, {(-1,0,-1),-96}, {(-2,1,-1),-16}, {(-2,-2,0),-216}, {(-\
1,-2,0),192}, {(0,-2,0),-36}, {(-2,-1,0),192}, {(-1,-1,0),-96}, {(\
-2,0,0),-36}, {(-2,-2,1),32}, {(-1,-2,1),-16}, {(-2,-1,1),-16}, {(\
-2,-2,2),-1}, {(2,2,2),1};

                         Dy du/dy = Tfy +

ky1 Ty1 + ky2 Ty2 + ky3 Ty3 + ky4 Ty4 + ky5 Ty5 + ky6 Ty6 +

    ky7 Ty7 + ky8 Ty8 + ky9 Ty9 + ky10 Ty10 + ky11 Ty11 +

    ky12 Ty12 + ky13 Ty13 + ky14 Ty14 + ky15 Ty15 + ky16 Ty16 +

    ky17 Ty17 + ky18 Ty18 + ky19 Ty19 + ky20 Ty20 + ky21 Ty21 +

    ky22 Ty22 + ky23 Ty23 + ky24 Ty24 + ky25 Ty25 + ky26 Ty26 +

    ky27 Ty27 + ky28 Ty28 + ky29 Ty29 + ky30 Ty30 + ky31 Ty31 +

    ky32 Ty32 + ky33 Ty33 + ky34 Ty34 + ky35 Ty35 + ky36 Ty36 +

    ky37 Ty37 + ky38 Ty38 + ky39 Ty39 + ky40 Ty40 + ky41 Ty41 +

    ky42 Ty42 + ky43 Ty43 + ky44 Ty44 + ky45 Ty45 + ky46 Ty46 +

    ky47 Ty47 + ky48 Ty48 + ky49 Ty49 + ky50 Ty50 + ky51 Ty51 +

    ky52 Ty52 + ky53 Ty53 + ky54 Ty54 + ky55 Ty55 + ky56 Ty56 +

    ky57 Ty57 + ky58 Ty58 + ky59 Ty59 + ky60 Ty60 + ky61 Ty61 +

    ky62 Ty62 + ky63 Ty63 + ky64 Ty64 + ky65 Ty65 + ky66 Ty66 +

    ky67 Ty67 + ky68 Ty68 + ky69 Ty69 + ky70 Ty70 + ky71 Ty71 +
```

```
ky72 Ty72 + ky73 Ty73 + ky74 Ty74 + ky75 Ty75 + ky76 Ty76 +

ky77 Ty77 + ky78 Ty78 + ky79 Ty79 + ky80 Ty80 + ky81 Ty81 +

ky82 Ty82 + ky83 Ty83 + ky84 Ty84 + ky85 Ty85 + ky86 Ty86 +

ky87 Ty87 + ky88 Ty88 + ky89 Ty89 + ky90 Ty90,   where

Tfy = {(-2,-2,-2),77/12}, {(-1,-2,-2),-26/3}, {(0,-2,-2),5/2}, {(-\
2,-1,-2),-38/3}, {(-1,-1,-2),16}, {(0,-1,-2),-4}, {(-2,0,-2),7}, {\
(-1,0,-2),-8}, {(0,0,-2),3/2}, {(-2,1,-2),-2/3}, {(-1,1,-2),2/3}, \
{(-2,2,-2),-1/12}, {(-2,-2,-1),-26/3}, {(-1,-2,-1),10}, {(0,-2,-1)\
,-2}, {(-2,-1,-1),16}, {(-1,-1,-1),-16}, {(0,-1,-1),2}, {(-2,0,-1)\
,-8}, {(-1,0,-1),6}, {(-2,1,-1),2/3}, {(-2,-2,0),5/2}, {(-1,-2,0),\
-2}, {(-2,-1,0),-4}, {(-1,-1,0),2}, {(-2,0,0),3/2};

Ty1 = {(-2,-2,-2),-1}, {(-1,-2,-2),4}, {(0,-2,-2),-6}, {(1,-2,-2),\
4}, {(2,-2,-2),-1}, {(-2,-1,-2),1}, {(-1,-1,-2),-4}, {(0,-1,-2),6}\
, {(1,-1,-2),-4}, {(2,-1,-2),1};
```

...

Some of the output is eliminated, since the stencils $Tyi = Txi$ for $i = 1..90$.

...

```
Ty89 = {(-2,-2,-2),-210}, {(-1,-2,-2),252}, {(0,-2,-2),-84}, {(1,-\
2,-2),7}, {(-2,-1,-2),336}, {(-1,-1,-2),-336}, {(0,-1,-2),84}, {(1\
,-1,-2),-4}, {(-2,0,-2),-168}, {(-1,0,-2),126}, {(0,0,-2),-18}, {(\
-2,1,-2),28}, {(-1,1,-2),-12}, {(-2,2,-2),-1}, {(-2,-2,-1),336}, {\
(-1,-2,-1),-336}, {(0,-2,-1),84}, {(1,-2,-1),-4}, {(-2,-1,-1),-448\
}, {(-1,-1,-1),336}, {(0,-1,-1),-48}, {(-2,0,-1),168}, {(-1,0,-1),\
-72}, {(-2,1,-1),-16}, {(-2,-2,0),-168}, {(-1,-2,0),126}, {(0,-2,0\
),-18}, {(-2,-1,0),168}, {(-1,-1,0),-72}, {(-2,0,0),-36}, {(-2,-2,\
1),28}, {(-1,-2,1),-12}, {(-2,-1,1),-16}, {(-2,-2,2),-1}, {(1,2,2)\
,1};

Ty90 = {(-2,-2,-2),-330}, {(-1,-2,-2),480}, {(0,-2,-2),-216}, {(1,\
-2,-2),32}, {(2,-2,-2),-1}, {(-2,-1,-2),480}, {(-1,-1,-2),-576}, {\
```

```
(0,-1,-2),192}, {(1,-1,-2),-16}, {(-2,0,-2),-216}, {(-1,0,-2),192}\
, {(0,0,-2),-36}, {(-2,1,-2),32}, {(-1,1,-2),-16}, {(-2,2,-2),-1},\
 {(-2,-2,-1),480}, {(-1,-2,-1),-576}, {(0,-2,-1),192}, {(1,-2,-1),\
-16}, {(-2,-1,-1),-576}, {(-1,-1,-1),512}, {(0,-1,-1),-96}, {(-2,0\
,-1),192}, {(-1,0,-1),-96}, {(-2,1,-1),-16}, {(-2,-2,0),-216}, {(-\
1,-2,0),192}, {(0,-2,0),-36}, {(-2,-1,0),192}, {(-1,-1,0),-96}, {(\
-2,0,0),-36}, {(-2,-2,1),32}, {(-1,-2,1),-16}, {(-2,-1,1),-16}, {(\
-2,-2,2),-1}, {(2,2,2),1};
```

Dz du/dz = Tfz +

```
kz1 Tz1 + kz2 Tz2 + kz3 Tz3 + kz4 Tz4 + kz5 Tz5 + kz6 Tz6 +
    kz7 Tz7 + kz8 Tz8 + kz9 Tz9 + kz10 Tz10 + kz11 Tz11 +
    kz12 Tz12 + kz13 Tz13 + kz14 Tz14 + kz15 Tz15 + kz16 Tz16 +
    kz17 Tz17 + kz18 Tz18 + kz19 Tz19 + kz20 Tz20 + kz21 Tz21 +
    kz22 Tz22 + kz23 Tz23 + kz24 Tz24 + kz25 Tz25 + kz26 Tz26 +
    kz27 Tz27 + kz28 Tz28 + kz29 Tz29 + kz30 Tz30 + kz31 Tz31 +
    kz32 Tz32 + kz33 Tz33 + kz34 Tz34 + kz35 Tz35 + kz36 Tz36 +
    kz37 Tz37 + kz38 Tz38 + kz39 Tz39 + kz40 Tz40 + kz41 Tz41 +
    kz42 Tz42 + kz43 Tz43 + kz44 Tz44 + kz45 Tz45 + kz46 Tz46 +
    kz47 Tz47 + kz48 Tz48 + kz49 Tz49 + kz50 Tz50 + kz51 Tz51 +
    kz52 Tz52 + kz53 Tz53 + kz54 Tz54 + kz55 Tz55 + kz56 Tz56 +
    kz57 Tz57 + kz58 Tz58 + kz59 Tz59 + kz60 Tz60 + kz61 Tz61 +
    kz62 Tz62 + kz63 Tz63 + kz64 Tz64 + kz65 Tz65 + kz66 Tz66 +
    kz67 Tz67 + kz68 Tz68 + kz69 Tz69 + kz70 Tz70 + kz71 Tz71 +
    kz72 Tz72 + kz73 Tz73 + kz74 Tz74 + kz75 Tz75 + kz76 Tz76 +
```

```
kz77 Tz77 + kz78 Tz78 + kz79 Tz79 + kz80 Tz80 + kz81 Tz81 +

kz82 Tz82 + kz83 Tz83 + kz84 Tz84 + kz85 Tz85 + kz86 Tz86 +

kz87 Tz87 + kz88 Tz88 + kz89 Tz89 + kz90 Tz90,  where

Tfz = {(-2,-2,-2),77/12}, {(-1,-2,-2),-26/3}, {(0,-2,-2),5/2}, {(-\
2,-1,-2),-26/3}, {(-1,-1,-2),10}, {(0,-1,-2),-2}, {(-2,0,-2),5/2},\
 {(-1,0,-2),-2}, {(-2,-2,-1),-38/3}, {(-1,-2,-1),16}, {(0,-2,-1),-\
4}, {(-2,-1,-1),16}, {(-1,-1,-1),-16}, {(0,-1,-1),2}, {(-2,0,-1),-\
4}, {(-1,0,-1),2}, {(-2,-2,0),7}, {(-1,-2,0),-8}, {(0,-2,0),3/2}, \
{(-2,-1,0),-8}, {(-1,-1,0),6}, {(-2,0,0),3/2}, {(-2,-2,1),-2/3}, {\
(-1,-2,1),2/3}, {(-2,-1,1),2/3}, {(-2,-2,2),-1/12};

Tz1 = {(-2,-2,-2),-1}, {(-1,-2,-2),4}, {(0,-2,-2),-6}, {(1,-2,-2),\
4}, {(2,-2,-2),-1}, {(-2,-1,-2),1}, {(-1,-1,-2),-4}, {(0,-1,-2),6}\
, {(1,-1,-2),-4}, {(2,-1,-2),1};
```

...

Some of the output is eliminated, since the stencils $Tzi = Txi$ for $i = 1..90$.

...

```
Tz89 = {(-2,-2,-2),-210}, {(-1,-2,-2),252}, {(0,-2,-2),-84}, {(1,-\
2,-2),7}, {(-2,-1,-2),336}, {(-1,-1,-2),-336}, {(0,-1,-2),84}, {(1\
,-1,-2),-4}, {(-2,0,-2),-168}, {(-1,0,-2),126}, {(0,0,-2),-18}, {(\
-2,1,-2),28}, {(-1,1,-2),-12}, {(-2,2,-2),-1}, {(-2,-2,-1),336}, {\
(-1,-2,-1),-336}, {(0,-2,-1),84}, {(1,-2,-1),-4}, {(-2,-1,-1),-448\
}, {(-1,-1,-1),336}, {(0,-1,-1),-48}, {(-2,0,-1),168}, {(-1,0,-1),\
-72}, {(-2,1,-1),-16}, {(-2,-2,0),-168}, {(-1,-2,0),126}, {(0,-2,0\
),-18}, {(-2,-1,0),168}, {(-1,-1,0),-72}, {(-2,0,0),-36}, {(-2,-2,\
1),28}, {(-1,-2,1),-12}, {(-2,-1,1),-16}, {(-2,-2,2),-1}, {(1,2,2)\
,1};

Tz90 = {(-2,-2,-2),-330}, {(-1,-2,-2),480}, {(0,-2,-2),-216}, {(1,\
-2,-2),32}, {(2,-2,-2),-1}, {(-2,-1,-2),480}, {(-1,-1,-2),-576}, {\
(0,-1,-2),192}, {(1,-1,-2),-16}, {(-2,0,-2),-216}, {(-1,0,-2),192}\
, {(0,0,-2),-36}, {(-2,1,-2),32}, {(-1,1,-2),-16}, {(-2,2,-2),-1},\
```

{(−2,−2,−1),480}, {(−1,−2,−1),−576}, {(0,−2,−1),192}, {(1,−2,−1),\
−16}, {(−2,−1,−1),−576}, {(−1,−1,−1),512}, {(0,−1,−1),−96}, {(−2,0\
,−1),192}, {(−1,0,−1),−96}, {(−2,1,−1),−16}, {(−2,−2,0),−216}, {(−\
1,−2,0),192}, {(0,−2,0),−36}, {(−2,−1,0),192}, {(−1,−1,0),−96}, {(\
−2,0,0),−36}, {(−2,−2,1),32}, {(−1,−2,1),−16}, {(−2,−1,1),−16}, {(\
−2,−2,2),−1}, {(2,2,2),1};

The grid−aligned one−dimensional solution which is excluded from the patent is given by

kx1 = 0, kx2 = 0, kx3 = 0, kx4 = 0, kx5 = 0, kx6 = 0, kx7 = 0, kx8\
 = 0, kx9 = 0, kx10 = 0, kx11 = 0, kx20 = 0, kx12 = 0, kx21 = 0, k\
x30 = 0, kx13 = 0, kx22 = 0, kx31 = −2/3, kx40 = 0, kx14 = 0, kx23\
 = 0, kx32 = 0, kx41 = 0, kx50 = 0, kx15 = 0, kx24 = 0, kx33 = 2/3\
, kx42 = 0, kx51 = 0, kx60 = 0, kx16 = 0, kx25 = 0, kx34 = −1/12, \
kx43 = 0, kx52 = 0, kx61 = 0, kx70 = 0, kx17 = 0, kx26 = 0, kx35 =\
 0, kx44 = 0, kx53 = 0, kx62 = 0, kx71 = 0, kx80 = 0, kx18 = 0, kx\
27 = 0, kx36 = 0, kx45 = 0, kx54 = 0, kx63 = 0, kx72 = 0, kx81 = 0\
, kx90 = 0, kx19 = 0, kx28 = 0, kx37 = 0, kx46 = 0, kx55 = 0, kx64\
 = 0, kx73 = 0, kx82 = 0, kx29 = 0, kx38 = 0, kx47 = 0, kx56 = 0, \
kx65 = 0, kx74 = 0, kx83 = 0, kx39 = 0, kx48 = 0, kx57 = 0, kx66 =\
 0, kx75 = 0, kx84 = 0, kx49 = 0, kx58 = 0, kx67 = 0, kx76 = 0, kx\
85 = 0, kx59 = 0, kx68 = 0, kx77 = 0, kx86 = 0, kx69 = 0, kx78 = 0\
, kx87 = 0, kx79 = 0, kx88 = 0, kx89 = 0

and

ky1 = 0, ky2 = 0, ky3 = 0, ky4 = 0, ky5 = 0, ky6 = 0, ky7 = 0, ky8\
 = 0, ky9 = 0, ky10 = 0, ky11 = 0, ky20 = 0, ky12 = 0, ky21 = 0, k\
y30 = 0, ky13 = 0, ky22 = 0, ky31 = 0, ky40 = 0, ky14 = 0, ky23 = \
0, ky32 = 0, ky41 = 0, ky50 = 0, ky15 = 0, ky24 = 0, ky33 = 0, ky4\
2 = −1/12, ky51 = 0, ky60 = 0, ky16 = 0, ky25 = 0, ky34 = 0, ky43 \
= 0, ky52 = 0, ky61 = 0, ky70 = 0, ky17 = 0, ky26 = 0, ky35 = 0, k\
y44 = 0, ky53 = 0, ky62 = 0, ky71 = 0, ky80 = 0, ky18 = 0, ky27 = \
0, ky36 = 0, ky45 = 0, ky54 = 0, ky63 = 0, ky72 = 0, ky81 = 0, ky9\
0 = 0, ky19 = 0, ky28 = −2/3, ky37 = 2/3, ky46 = 0, ky55 = 0, ky64\
 = 0, ky73 = 0, ky82 = 0, ky29 = 0, ky38 = 0, ky47 = 0, ky56 = 0, \
ky65 = 0, ky74 = 0, ky83 = 0, ky39 = 0, ky48 = 0, ky57 = 0, ky66 =\
 0, ky75 = 0, ky84 = 0, ky49 = 0, ky58 = 0, ky67 = 0, ky76 = 0, ky\
85 = 0, ky59 = 0, ky68 = 0, ky77 = 0, ky86 = 0, ky69 = 0, ky78 = 0\

```
, ky87 = 0, ky79 = 0, ky88 = 0, ky89 = 0
```

and

```
kz1 = 0, kz2 = 0, kz3 = 0, kz4 = 0, kz5 = 0, kz6 = 0, kz7 = 0, kz8\
 = 0, kz9 = 0, kz10 = 0, kz11 = 0, kz20 = 0, kz12 = 0, kz21 = 0, k\
z30 = 0, kz13 = 0, kz22 = 0, kz31 = 0, kz40 = 0, kz14 = -2/3, kz23\
 = 0, kz32 = 0, kz41 = 0, kz50 = 0, kz15 = 0, kz24 = 0, kz33 = 0, \
kz42 = 0, kz51 = 0, kz60 = 0, kz16 = 0, kz25 = 0, kz34 = 0, kz43 =\
 0, kz52 = 0, kz61 = 0, kz70 = 0, kz17 = 0, kz26 = 0, kz35 = 0, kz\
44 = 0, kz53 = 0, kz62 = 0, kz71 = 0, kz80 = 0, kz18 = 0, kz27 = 0\
, kz36 = 0, kz45 = 0, kz54 = 2/3, kz63 = 0, kz72 = 0, kz81 = 0, kz\
90 = 0, kz19 = 0, kz28 = 0, kz37 = 0, kz46 = 0, kz55 = 0, kz64 = 0\
, kz73 = 0, kz82 = 0, kz29 = 0, kz38 = 0, kz47 = 0, kz56 = 0, kz65\
 = 0, kz74 = 0, kz83 = 0, kz39 = 0, kz48 = 0, kz57 = 0, kz66 = 0, \
kz75 = 0, kz84 = 0, kz49 = 0, kz58 = 0, kz67 = 0, kz76 = 0, kz85 =\
 0, kz59 = 0, kz68 = 0, kz77 = 0, kz86 = 0, kz69 = 0, kz78 = -1/12\
, kz87 = 0, kz79 = 0, kz88 = 0, kz89 = 0
```

APPENDIX 10 : THE OUTPUT FOR D_{100}, ORDER 4, ON THE GRID $(-2..2)^3$, OPTIMIZE=1

The first part of this output is the same as of Appendix 9. It is ended by :

...

```
We use the notation
a for the angle alpha, the rotation around the grid z-axis,
b for the angle beta,  the rotation around the new y'-axis,
c for the angle gamma, the rotation around newest x''-axis,
which is the axis e1 of the local basis B, as explained in the
patent application.
The present optimization imposes constraints on the error terms.
The error terms in the local basis are written to the files
direrror.mu and direrror.txt.
The notation Epqr means the error in the approximation related to
the mixed pth derivative w.r.t. direction e1,
the qth derivative w.r.t. to e2 and the rth derivative w.r.t. e3
e.g. E201 is the contribution related to the mixed 2nd derivative
along axis e1, and the first derivative along axis e3

                    The error term E005 = 0

                    The error term E014 = 0

                    The error term E023 = 0

                    The error term E032 = 0

                    The error term E041 = 0

                    The error term E050 = 0
```

The error terms $E005$ etc. are stored upon request in separate files. As an example, the program generates the file `D_100-O_4-G_-22-OP_1/direrror005.mu` which contains just this error, which is assigned to the variable direrrtmp. The file contains some 9595 lines, starting with

```
sysassign(direrrtmp, hold(1/30*Dx^4*cos(a)*cos(b)*sin(a)^5*sin(c)^\
5 - 1/30*Dz^4*cos(b)^5*cos(c)^5*sin(b) - 1/30*Dy^4*cos(a)^5*cos(b)\
*sin(a)*sin(c)^5 + Dx^4*ky1*cos(a)*cos(b)*sin(a)^5*sin(c)^5 + 2*Dx\
^4*ky3*cos(a)*cos(b)*sin(a)^5*sin(c)^5 + 3*Dx^4*ky6*cos(a)*cos(b)*\
sin(a)^5*sin(c)^5 - Dy^4*kx7*cos(a)^5*cos(b)*sin(a)*sin(c)^5 - 2*D\
y^4*kx8*cos(a)^5*cos(b)*sin(a)*sin(c)^5 - 3*Dy^4*kx9*cos(a)^5*cos(\
b)*sin(a)*sin(c)^5 + 4*Dx^4*ky10*cos(a)*cos(b)*sin(a)^5*sin(c)^5 -\
 4*Dy^4*kx10*cos(a)^5*cos(b)*sin(a)*sin(c)^5 + 3*Dx^4*ky20*cos(a)*\
cos(b)*sin(a)^5*sin(c)^5 + Dx^4*ky30*cos(a)*cos(b)*sin(a)^5*sin(c)\
^5 + Dx^4*ky13*cos(a)*cos(b)*sin(a)^5*sin(c)^5 - Dy^4*kx22*cos(a)^\
5*cos(b)*sin(a)*sin(c)^5 - 2*Dy^4*kx23*cos(a)^5*cos(b)*sin(a)*sin(\
c)^5 - Dy^4*kx41*cos(a)^5*cos(b)*sin(a)*sin(c)^5 + Dx^4*ky51*cos(a\
)*cos(b)*sin(a)^5*sin(c)^5 - 3*Dy^4*kx24*cos(a)^5*cos(b)*sin(a)*si\
n(c)^5 - 2*Dy^4*kx42*cos(a)^5*cos(b)*sin(a)*sin(c)^5 + 2*Dx^4*ky16\
*cos(a)*cos(b)*sin(a)^5*sin(c)^5 + 4*Dx^4*ky25*cos(a)*cos(b)*sin(a\
)^5*sin(c)^5 + 2*Dx^4*ky34*cos(a)*cos(b)*sin(a)^5*sin(c)^5 + 3*Dx^\
4*ky61*cos(a)*cos(b)*sin(a)^5*sin(c)^5 - 4*Dy^4*kx25*cos(a)^5*cos(\
b)*sin(a)*sin(c)^5 - 3*Dy^4*kx43*cos(a)^5*cos(b)*sin(a)*sin(c)^5 +\
```

... and ending with ...

```
 + 12*Dx^4/Dy*Dz*ky39*cos(a)^2*cos(b)^2*cos(c)^3*sin(a)^3*sin(b)^2\
*sin(c)^2 + 18*Dx^4/Dy*Dz*ky66*cos(a)^2*cos(b)^2*cos(c)^3*sin(a)^3\
*sin(b)^2*sin(c)^2 + 24*Dx^4/Dy*Dz*ky75*cos(a)^2*cos(b)^2*cos(c)^3\
*sin(a)^3*sin(b)^2*sin(c)^2 + 9/Dx*Dy^3*Dz^2*kx59*cos(a)^2*cos(b)^\
3*cos(c)^4*sin(a)^2*sin(b)^2*sin(c) + 72/Dx*Dy^3*Dz^2*kx86*cos(a)^\
2*cos(b)^3*cos(c)^4*sin(a)^2*sin(b)^2*sin(c) + 24/Dx*Dy^4*Dz*kx86*\
cos(a)^3*cos(b)^2*cos(c)^3*sin(a)^2*sin(b)^2*sin(c)^2 - 72*Dx^3/Dy\
*Dz^2*ky85*cos(a)^2*cos(b)^3*cos(c)^4*sin(a)^2*sin(b)^2*sin(c) + 2\
4*Dx^4/Dy*Dz*ky85*cos(a)^2*cos(b)^2*cos(c)^3*sin(a)^3*sin(b)^2*sin\
(c)^2 + 72/Dx*Dy^3*Dz^2*kx87*cos(a)^2*cos(b)^3*cos(c)^4*sin(a)^2*s\
in(b)^2*sin(c) + 24/Dx*Dy^4*Dz*kx87*cos(a)^3*cos(b)^2*cos(c)^3*sin\
(a)^2*sin(b)^2*sin(c)^2 + 72/Dx*Dy^3*Dz^2*kx88*cos(a)^2*cos(b)^3*c\
os(c)^4*sin(a)^2*sin(b)^2*sin(c) + 24/Dx*Dy^4*Dz*kx88*cos(a)^3*cos\
(b)^2*cos(c)^3*sin(a)^2*sin(b)^2*sin(c)^2 - 18*Dx^3/Dy*Dz^2*ky69*c\
os(a)^2*cos(b)^3*cos(c)^4*sin(a)^2*sin(b)^2*sin(c) + 72/Dx*Dy^3*Dz\
^2*kx89*cos(a)^2*cos(b)^3*cos(c)^4*sin(a)^2*sin(b)^2*sin(c) + 24/D\
x*Dy^4*Dz*kx89*cos(a)^3*cos(b)^2*cos(c)^3*sin(a)^2*sin(b)^2*sin(c)\
^2 - 18*Dx^3/Dy*Dz^2*ky79*cos(a)^2*cos(b)^3*cos(c)^4*sin(a)^2*sin(\
b)^2*sin(c) - 18*Dx^3/Dy*Dz^2*ky89*cos(a)^2*cos(b)^3*cos(c)^4*sin(\
a)^2*sin(b)^2*sin(c))):
```

Appendix 11 : The output for D_{100}, Order 4, on the grid $(-2..2)^3$, optimize=2

```
                    ----- preparations -----

                  gridmin = -2, gridmax = 2

                  se1 = 1, se2 = 0, se3 = 0

                          order = 4

                        optimize = 2

                          check = 0

                    ----- setup equations -----

     Discretization of a derivative in 3D with order 4 :
     1 times a derivation along the direction e1,
     0 times a derivation along the direction e2,
     0 times a derivation along the direction e3,
     the highest derivative is 1

 Establishing the equations for a consistent discretization.

         Computing the local derivative error terms.

     Equations for optimizing a pure derivative along e1

              The equations are ready to be solved.

                  the time spent is 5.77 s.

                    ----- solve equations -----

     Solving the equations for consistent approximations

        Solving the directional equations - along x1
```

Solving the directional equations – along diag x1–x2

Solving the directional equations – along diag x1–x2–x3

The solution is written to the file : D_100–O_4–G_–22–OP_2/soln.mu

the time spent is 1441.38 s.

----- analyze solution ------

The constraints specified in the file generate–discretization,

gridmin = –2, gridmax = 2,

se1 = 1, se2 = 0, se3 = 0,

order = 4,

optimize = 2,

check = 0,

are satisfied by the following solution, which is written using the notation k... for a free coefficient and where T... represents a stencil which is given by a list of nodes with corresponding weights in the notation {(i,j,k),w} where the node is given by the indices i,j,k and the weight by w, and where Dx, Dy and Dz are the grid spacings in the three coordinate directions, in the following approximations : (a line which ends with \ is continued on the next line)

Dx du/dx = Tfx +

kx1 Tx1 + kx2 Tx2 + kx3 Tx3 + kx4 Tx4 + kx5 Tx5 + kx6 Tx6 +

kx7 Tx7 + kx8 Tx8 + kx9 Tx9 + kx10 Tx10 + kx11 Tx11 +

kx12 Tx12 + kx13 Tx13 + kx14 Tx14 + kx15 Tx15 + kx16 Tx16 +

```
kx17 Tx17 + kx18 Tx18 + kx19 Tx19 + kx20 Tx20 + kx21 Tx21 +
kx22 Tx22 + kx23 Tx23 + kx24 Tx24 + kx25 Tx25 + kx26 Tx26 +
kx27 Tx27 + kx28 Tx28 + kx29 Tx29 + kx30 Tx30 + kx31 Tx31 +
kx32 Tx32 + kx33 Tx33 + kx34 Tx34 + kx35 Tx35 + kx36 Tx36 +
kx37 Tx37 + kx38 Tx38 + kx39 Tx39 + kx40 Tx40 + kx41 Tx41 +
kx42 Tx42 + kx43 Tx43 + kx44 Tx44 + kx45 Tx45 + kx46 Tx46 +
kx47 Tx47 + kx48 Tx48 + kx49 Tx49 + kx50 Tx50 + kx51 Tx51 +
kx52 Tx52 + kx53 Tx53 + kx54 Tx54 + kx55 Tx55 + kx56 Tx56 +
kx57 Tx57 + kx58 Tx58 + kx59 Tx59 + kx60 Tx60 + kx61 Tx61 +
kx62 Tx62 + kx63 Tx63 + kx64 Tx64 + kx65 Tx65 + kx66 Tx66 +
kx67 Tx67 + kx68 Tx68 + kx69 Tx69 + kx70 Tx70 + kx71 Tx71 +
kx72 Tx72,  where

Tfx = {(-2,-2,-2),-9/4}, {(-1,-2,-2),5}, {(0,-2,-2),-3}, {(2,-2,-2\
),1/4}, {(-2,-1,-2),25/6}, {(-1,-1,-2),-28/3}, {(0,-1,-2),6}, {(1,\
-1,-2),-2/3}, {(2,-1,-2),-1/6}, {(-2,0,-2),-11/6}, {(-1,0,-2),4}, \
{(0,0,-2),-5/2}, {(1,0,-2),1/3}, {(-2,-2,-1),25/6}, {(-1,-2,-1),-2\
8/3}, {(0,-2,-1),6}, {(1,-2,-1),-2/3}, {(2,-2,-1),-1/6}, {(-2,-1,-\
1),-22/3}, {(-1,-1,-1),16}, {(0,-1,-1),-10}, {(1,-1,-1),4/3}, {(-2\
,0,-1),3}, {(-1,0,-1),-6}, {(0,0,-1),3}, {(-2,-2,0),-11/6}, {(-1,-\
2,0),4}, {(0,-2,0),-5/2}, {(1,-2,0),1/3}, {(-2,-1,0),3}, {(-1,-1,0\
),-6}, {(0,-1,0),3}, {(-2,0,0),-1}, {(-1,0,0),1};

Tx1 = {(-2,-2,-2),1}, {(-1,-2,-2),-4}, {(0,-2,-2),6}, {(1,-2,-2),-\
4}, {(2,-2,-2),1}, {(-2,-1,-2),-2}, {(-1,-1,-2),8}, {(0,-1,-2),-12\
}, {(1,-1,-2),8}, {(2,-1,-2),-2}, {(-2,0,-2),1}, {(-1,0,-2),-4}, {\
(0,0,-2),6}, {(1,0,-2),-4}, {(2,0,-2),1};

Tx2 = {(-2,-2,-2),1}, {(-1,-2,-2),-3}, {(0,-2,-2),3}, {(1,-2,-2),-\
```

```
1}, {(-2,-1,-2),-3}, {(-1,-1,-2),9}, {(0,-1,-2),-9}, {(1,-1,-2),3}\
, {(-2,0,-2),3}, {(-1,0,-2),-9}, {(0,0,-2),9}, {(1,0,-2),-3}, {(-2\
,1,-2),-1}, {(-1,1,-2),3}, {(0,1,-2),-3}, {(1,1,-2),1};

Tx3 = {(-2,-2,-2),6}, {(-1,-2,-2),-20}, {(0,-2,-2),24}, {(1,-2,-2)\
,-12}, {(2,-2,-2),2}, {(-2,-1,-2),-15}, {(-1,-1,-2),48}, {(0,-1,-2\
),-54}, {(1,-1,-2),24}, {(2,-1,-2),-3}, {(-2,0,-2),12}, {(-1,0,-2)\
,-36}, {(0,0,-2),36}, {(1,0,-2),-12}, {(-2,1,-2),-3}, {(-1,1,-2),8\
}, {(0,1,-2),-6}, {(2,1,-2),1};

Tx4 = {(-2,-2,-2),1}, {(-1,-2,-2),-2}, {(0,-2,-2),1}, {(-2,-1,-2),\
-4}, {(-1,-1,-2),8}, {(0,-1,-2),-4}, {(-2,0,-2),6}, {(-1,0,-2),-12\
}, {(0,0,-2),6}, {(-2,1,-2),-4}, {(-1,1,-2),8}, {(0,1,-2),-4}, {(-\
2,2,-2),1}, {(-1,2,-2),-2}, {(0,2,-2),1};

Tx5 = {(-2,-2,-2),6}, {(-1,-2,-2),-15}, {(0,-2,-2),12}, {(1,-2,-2)\
,-3}, {(-2,-1,-2),-20}, {(-1,-1,-2),48}, {(0,-1,-2),-36}, {(1,-1,-\
2),8}, {(-2,0,-2),24}, {(-1,0,-2),-54}, {(0,0,-2),36}, {(1,0,-2),-\
6}, {(-2,1,-2),-12}, {(-1,1,-2),24}, {(0,1,-2),-12}, {(-2,2,-2),2}\
, {(-1,2,-2),-3}, {(1,2,-2),1};

Tx6 = {(-2,-2,-2),21}, {(-1,-2,-2),-60}, {(0,-2,-2),60}, {(1,-2,-2\
),-24}, {(2,-2,-2),3}, {(-2,-1,-2),-60}, {(-1,-1,-2),160}, {(0,-1,\
-2),-144}, {(1,-1,-2),48}, {(2,-1,-2),-4}, {(-2,0,-2),60}, {(-1,0,\
-2),-144}, {(0,0,-2),108}, {(1,0,-2),-24}, {(-2,1,-2),-24}, {(-1,1\
,-2),48}, {(0,1,-2),-24}, {(-2,2,-2),3}, {(-1,2,-2),-4}, {(2,2,-2)\
,1};

Tx7 = {(-2,-2,-2),1}, {(-1,-2,-2),-4}, {(0,-2,-2),6}, {(1,-2,-2),-\
4}, {(2,-2,-2),1}, {(-2,-1,-2),-1}, {(-1,-1,-2),4}, {(0,-1,-2),-6}\
, {(1,-1,-2),4}, {(2,-1,-2),-1}, {(-2,-2,-1),-1}, {(-1,-2,-1),4}, \
{(0,-2,-1),-6}, {(1,-2,-1),4}, {(2,-2,-1),-1}, {(-2,-1,-1),1}, {(-\
1,-1,-1),-4}, {(0,-1,-1),6}, {(1,-1,-1),-4}, {(2,-1,-1),1};

Tx8 = {(-2,-2,-2),1}, {(-1,-2,-2),-3}, {(0,-2,-2),3}, {(1,-2,-2),-\
1}, {(-2,-1,-2),-2}, {(-1,-1,-2),6}, {(0,-1,-2),-6}, {(1,-1,-2),2}\
, {(-2,0,-2),1}, {(-1,0,-2),-3}, {(0,0,-2),3}, {(1,0,-2),-1}, {(-2\
,-2,-1),-1}, {(-1,-2,-1),3}, {(0,-2,-1),-3}, {(1,-2,-1),1}, {(-2,-\
1,-1),2}, {(-1,-1,-1),-6}, {(0,-1,-1),6}, {(1,-1,-1),-2}, {(-2,0,-\
1),-1}, {(-1,0,-1),3}, {(0,0,-1),-3}, {(1,0,-1),1};
```

```
Tx9 = {(-2,-2,-2),6}, {(-1,-2,-2),-20}, {(0,-2,-2),24}, {(1,-2,-2)\
,-12}, {(2,-2,-2),2}, {(-2,-1,-2),-10}, {(-1,-1,-2),32}, {(0,-1,-2\
),-36}, {(1,-1,-2),16}, {(2,-1,-2),-2}, {(-2,0,-2),4}, {(-1,0,-2),\
-12}, {(0,0,-2),12}, {(1,0,-2),-4}, {(-2,-2,-1),-5}, {(-1,-2,-1),1\
6}, {(0,-2,-1),-18}, {(1,-2,-1),8}, {(2,-2,-1),-1}, {(-2,-1,-1),8}\
, {(-1,-1,-1),-24}, {(0,-1,-1),24}, {(1,-1,-1),-8}, {(-2,0,-1),-3}\
, {(-1,0,-1),8}, {(0,0,-1),-6}, {(2,0,-1),1};

Tx10 = {(-2,-2,-2),1}, {(-1,-2,-2),-2}, {(0,-2,-2),1}, {(-2,-1,-2)\
,-3}, {(-1,-1,-2),6}, {(0,-1,-2),-3}, {(-2,0,-2),3}, {(-1,0,-2),-6\
}, {(0,0,-2),3}, {(-2,1,-2),-1}, {(-1,1,-2),2}, {(0,1,-2),-1}, {(-\
2,-2,-1),-1}, {(-1,-2,-1),2}, {(0,-2,-1),-1}, {(-2,-1,-1),3}, {(-1\
,-1,-1),-6}, {(0,-1,-1),3}, {(-2,0,-1),-3}, {(-1,0,-1),6}, {(0,0,-\
1),-3}, {(-2,1,-1),1}, {(-1,1,-1),-2}, {(0,1,-1),1};

Tx11 = {(-2,-2,-2),6}, {(-1,-2,-2),-15}, {(0,-2,-2),12}, {(1,-2,-2\
),-3}, {(-2,-1,-2),-15}, {(-1,-1,-2),36}, {(0,-1,-2),-27}, {(1,-1,\
-2),6}, {(-2,0,-2),12}, {(-1,0,-2),-27}, {(0,0,-2),18}, {(1,0,-2),\
-3}, {(-2,1,-2),-3}, {(-1,1,-2),6}, {(0,1,-2),-3}, {(-2,-2,-1),-5}\
, {(-1,-2,-1),12}, {(0,-2,-1),-9}, {(1,-2,-1),2}, {(-2,-1,-1),12},\
 {(-1,-1,-1),-27}, {(0,-1,-1),18}, {(1,-1,-1),-3}, {(-2,0,-1),-9},\
 {(-1,0,-1),18}, {(0,0,-1),-9}, {(-2,1,-1),2}, {(-1,1,-1),-3}, {(1\
,1,-1),1};

Tx12 = {(-2,-2,-2),21}, {(-1,-2,-2),-60}, {(0,-2,-2),60}, {(1,-2,-\
2),-24}, {(2,-2,-2),3}, {(-2,-1,-2),-45}, {(-1,-1,-2),120}, {(0,-1\
,-2),-108}, {(1,-1,-2),36}, {(2,-1,-2),-3}, {(-2,0,-2),30}, {(-1,0\
,-2),-72}, {(0,0,-2),54}, {(1,0,-2),-12}, {(-2,1,-2),-6}, {(-1,1,-\
2),12}, {(0,1,-2),-6}, {(-2,-2,-1),-15}, {(-1,-2,-1),40}, {(0,-2,-\
1),-36}, {(1,-2,-1),12}, {(2,-2,-1),-1}, {(-2,-1,-1),30}, {(-1,-1,\
-1),-72}, {(0,-1,-1),54}, {(1,-1,-1),-12}, {(-2,0,-1),-18}, {(-1,0\
,-1),36}, {(0,0,-1),-18}, {(-2,1,-1),3}, {(-1,1,-1),-4}, {(2,1,-1)\
,1};

Tx13 = {(-2,-2,-2),1}, {(-1,-2,-2),-1}, {(-2,-1,-2),-4}, {(-1,-1,-\
2),4}, {(-2,0,-2),6}, {(-1,0,-2),-6}, {(-2,1,-2),-4}, {(-1,1,-2),4\
}, {(-2,2,-2),1}, {(-1,2,-2),-1}, {(-2,-2,-1),-1}, {(-1,-2,-1),1},\
 {(-2,-1,-1),4}, {(-1,-1,-1),-4}, {(-2,0,-1),-6}, {(-1,0,-1),6}, {\
(-2,1,-1),4}, {(-1,1,-1),-4}, {(-2,2,-1),-1}, {(-1,2,-1),1};
```

```
Tx14 = {(-2,-2,-2),6}, {(-1,-2,-2),-10}, {(0,-2,-2),4}, {(-2,-1,-2\
),-20}, {(-1,-1,-2),32}, {(0,-1,-2),-12}, {(-2,0,-2),24}, {(-1,0,-\
2),-36}, {(0,0,-2),12}, {(-2,1,-2),-12}, {(-1,1,-2),16}, {(0,1,-2)\
,-4}, {(-2,2,-2),2}, {(-1,2,-2),-2}, {(-2,-2,-1),-5}, {(-1,-2,-1),\
8}, {(0,-2,-1),-3}, {(-2,-1,-1),16}, {(-1,-1,-1),-24}, {(0,-1,-1),\
8}, {(-2,0,-1),-18}, {(-1,0,-1),24}, {(0,0,-1),-6}, {(-2,1,-1),8},\
 {(-1,1,-1),-8}, {(-2,2,-1),-1}, {(0,2,-1),1};

Tx15 = {(-2,-2,-2),21}, {(-1,-2,-2),-45}, {(0,-2,-2),30}, {(1,-2,-\
2),-6}, {(-2,-1,-2),-60}, {(-1,-1,-2),120}, {(0,-1,-2),-72}, {(1,-\
1,-2),12}, {(-2,0,-2),60}, {(-1,0,-2),-108}, {(0,0,-2),54}, {(1,0,\
-2),-6}, {(-2,1,-2),-24}, {(-1,1,-2),36}, {(0,1,-2),-12}, {(-2,2,-\
2),3}, {(-1,2,-2),-3}, {(-2,-2,-1),-15}, {(-1,-2,-1),30}, {(0,-2,-\
1),-18}, {(1,-2,-1),3}, {(-2,-1,-1),40}, {(-1,-1,-1),-72}, {(0,-1,\
-1),36}, {(1,-1,-1),-4}, {(-2,0,-1),-36}, {(-1,0,-1),54}, {(0,0,-1\
),-18}, {(-2,1,-1),12}, {(-1,1,-1),-12}, {(-2,2,-1),-1}, {(1,2,-1)\
,1};

Tx16 = {(-2,-2,-2),56}, {(-1,-2,-2),-140}, {(0,-2,-2),120}, {(1,-2\
,-2),-40}, {(2,-2,-2),4}, {(-2,-1,-2),-140}, {(-1,-1,-2),320}, {(0\
,-1,-2),-240}, {(1,-1,-2),64}, {(2,-1,-2),-4}, {(-2,0,-2),120}, {(\
-1,0,-2),-240}, {(0,0,-2),144}, {(1,0,-2),-24}, {(-2,1,-2),-40}, {\
(-1,1,-2),64}, {(0,1,-2),-24}, {(-2,2,-2),4}, {(-1,2,-2),-4}, {(-2\
,-2,-1),-35}, {(-1,-2,-1),80}, {(0,-2,-1),-60}, {(1,-2,-1),16}, {(\
2,-2,-1),-1}, {(-2,-1,-1),80}, {(-1,-1,-1),-160}, {(0,-1,-1),96}, \
{(1,-1,-1),-16}, {(-2,0,-1),-60}, {(-1,0,-1),96}, {(0,0,-1),-36}, \
{(-2,1,-1),16}, {(-1,1,-1),-16}, {(-2,2,-1),-1}, {(2,2,-1),1};

Tx17 = {(-2,-2,-2),1}, {(-1,-2,-2),-4}, {(0,-2,-2),6}, {(1,-2,-2),\
-4}, {(2,-2,-2),1}, {(-2,-2,-1),-2}, {(-1,-2,-1),8}, {(0,-2,-1),-1\
2}, {(1,-2,-1),8}, {(2,-2,-1),-2}, {(-2,-2,0),1}, {(-1,-2,0),-4}, \
{(0,-2,0),6}, {(1,-2,0),-4}, {(2,-2,0),1};

Tx18 = {(-2,-2,-2),1}, {(-1,-2,-2),-3}, {(0,-2,-2),3}, {(1,-2,-2),\
-1}, {(-2,-1,-2),-1}, {(-1,-1,-2),3}, {(0,-1,-2),-3}, {(1,-1,-2),1\
}, {(-2,-2,-1),-2}, {(-1,-2,-1),6}, {(0,-2,-1),-6}, {(1,-2,-1),2},\
 {(-2,-1,-1),2}, {(-1,-1,-1),-6}, {(0,-1,-1),6}, {(1,-1,-1),-2}, {\
(-2,-2,0),1}, {(-1,-2,0),-3}, {(0,-2,0),3}, {(1,-2,0),-1}, {(-2,-1\
,0),-1}, {(-1,-1,0),3}, {(0,-1,0),-3}, {(1,-1,0),1};
```

```
Tx19 = {(-2,-2,-2),6}, {(-1,-2,-2),-20}, {(0,-2,-2),24}, {(1,-2,-2\
),-12}, {(2,-2,-2),2}, {(-2,-1,-2),-5}, {(-1,-1,-2),16}, {(0,-1,-2\
),-18}, {(1,-1,-2),8}, {(2,-1,-2),-1}, {(-2,-2,-1),-10}, {(-1,-2,-\
1),32}, {(0,-2,-1),-36}, {(1,-2,-1),16}, {(2,-2,-1),-2}, {(-2,-1,-\
1),8}, {(-1,-1,-1),-24}, {(0,-1,-1),24}, {(1,-1,-1),-8}, {(-2,-2,0\
),4}, {(-1,-2,0),-12}, {(0,-2,0),12}, {(1,-2,0),-4}, {(-2,-1,0),-3\
}, {(-1,-1,0),8}, {(0,-1,0),-6}, {(2,-1,0),1};

Tx20 = {(-2,-2,-2),1}, {(-1,-2,-2),-2}, {(0,-2,-2),1}, {(-2,-1,-2)\
,-2}, {(-1,-1,-2),4}, {(0,-1,-2),-2}, {(-2,0,-2),1}, {(-1,0,-2),-2\
}, {(0,0,-2),1}, {(-2,-2,-1),-2}, {(-1,-2,-1),4}, {(0,-2,-1),-2}, \
{(-2,-1,-1),4}, {(-1,-1,-1),-8}, {(0,-1,-1),4}, {(-2,0,-1),-2}, {(\
-1,0,-1),4}, {(0,0,-1),-2}, {(-2,-2,0),1}, {(-1,-2,0),-2}, {(0,-2,\
0),1}, {(-2,-1,0),-2}, {(-1,-1,0),4}, {(0,-1,0),-2}, {(-2,0,0),1},\
 {(-1,0,0),-2}, {(0,0,0),1};

Tx21 = {(-2,-2,-2),6}, {(-1,-2,-2),-15}, {(0,-2,-2),12}, {(1,-2,-2\
),-3}, {(-2,-1,-2),-10}, {(-1,-1,-2),24}, {(0,-1,-2),-18}, {(1,-1,\
-2),4}, {(-2,0,-2),4}, {(-1,0,-2),-9}, {(0,0,-2),6}, {(1,0,-2),-1}\
, {(-2,-2,-1),-10}, {(-1,-2,-1),24}, {(0,-2,-1),-18}, {(1,-2,-1),4\
}, {(-2,-1,-1),16}, {(-1,-1,-1),-36}, {(0,-1,-1),24}, {(1,-1,-1),-\
4}, {(-2,0,-1),-6}, {(-1,0,-1),12}, {(0,0,-1),-6}, {(-2,-2,0),4}, \
{(-1,-2,0),-9}, {(0,-2,0),6}, {(1,-2,0),-1}, {(-2,-1,0),-6}, {(-1,\
-1,0),12}, {(0,-1,0),-6}, {(-2,0,0),2}, {(-1,0,0),-3}, {(1,0,0),1};

Tx22 = {(-2,-2,-2),21}, {(-1,-2,-2),-60}, {(0,-2,-2),60}, {(1,-2,-\
2),-24}, {(2,-2,-2),3}, {(-2,-1,-2),-30}, {(-1,-1,-2),80}, {(0,-1,\
-2),-72}, {(1,-1,-2),24}, {(2,-1,-2),-2}, {(-2,0,-2),10}, {(-1,0,-\
2),-24}, {(0,0,-2),18}, {(1,0,-2),-4}, {(-2,-2,-1),-30}, {(-1,-2,-\
1),80}, {(0,-2,-1),-72}, {(1,-2,-1),24}, {(2,-2,-1),-2}, {(-2,-1,-\
1),40}, {(-1,-1,-1),-96}, {(0,-1,-1),72}, {(1,-1,-1),-16}, {(-2,0,\
-1),-12}, {(-1,0,-1),24}, {(0,0,-1),-12}, {(-2,-2,0),10}, {(-1,-2,\
0),-24}, {(0,-2,0),18}, {(1,-2,0),-4}, {(-2,-1,0),-12}, {(-1,-1,0)\
,24}, {(0,-1,0),-12}, {(-2,0,0),3}, {(-1,0,0),-4}, {(2,0,0),1};

Tx23 = {(-2,-2,-2),1}, {(-1,-2,-2),-1}, {(-2,-1,-2),-3}, {(-1,-1,-\
2),3}, {(-2,0,-2),3}, {(-1,0,-2),-3}, {(-2,1,-2),-1}, {(-1,1,-2),1\
}, {(-2,-2,-1),-2}, {(-1,-2,-1),2}, {(-2,-1,-1),6}, {(-1,-1,-1),-6\
}, {(-2,0,-1),-6}, {(-1,0,-1),6}, {(-2,1,-1),2}, {(-1,1,-1),-2}, {\
```

```
(-2,-2,0),1}, {(-1,-2,0),-1}, {(-2,-1,0),-3}, {(-1,-1,0),3}, {(-2,\
0,0),3}, {(-1,0,0),-3}, {(-2,1,0),-1}, {(-1,1,0),1};

Tx24 = {(-2,-2,-2),6}, {(-1,-2,-2),-10}, {(0,-2,-2),4}, {(-2,-1,-2\
),-15}, {(-1,-1,-2),24}, {(0,-1,-2),-9}, {(-2,0,-2),12}, {(-1,0,-2\
),-18}, {(0,0,-2),6}, {(-2,1,-2),-3}, {(-1,1,-2),4}, {(0,1,-2),-1}\
, {(-2,-2,-1),-10}, {(-1,-2,-1),16}, {(0,-2,-1),-6}, {(-2,-1,-1),2\
4}, {(-1,-1,-1),-36}, {(0,-1,-1),12}, {(-2,0,-1),-18}, {(-1,0,-1),\
24}, {(0,0,-1),-6}, {(-2,1,-1),4}, {(-1,1,-1),-4}, {(-2,-2,0),4}, \
{(-1,-2,0),-6}, {(0,-2,0),2}, {(-2,-1,0),-9}, {(-1,-1,0),12}, {(0,\
-1,0),-3}, {(-2,0,0),6}, {(-1,0,0),-6}, {(-2,1,0),-1}, {(0,1,0),1};

Tx25 = {(-2,-2,-2),21}, {(-1,-2,-2),-45}, {(0,-2,-2),30}, {(1,-2,-\
2),-6}, {(-2,-1,-2),-45}, {(-1,-1,-2),90}, {(0,-1,-2),-54}, {(1,-1\
,-2),9}, {(-2,0,-2),30}, {(-1,0,-2),-54}, {(0,0,-2),27}, {(1,0,-2)\
,-3}, {(-2,1,-2),-6}, {(-1,1,-2),9}, {(0,1,-2),-3}, {(-2,-2,-1),-3\
0}, {(-1,-2,-1),60}, {(0,-2,-1),-36}, {(1,-2,-1),6}, {(-2,-1,-1),6\
0}, {(-1,-1,-1),-108}, {(0,-1,-1),54}, {(1,-1,-1),-6}, {(-2,0,-1),\
-36}, {(-1,0,-1),54}, {(0,0,-1),-18}, {(-2,1,-1),6}, {(-1,1,-1),-6\
}, {(-2,-2,0),10}, {(-1,-2,0),-18}, {(0,-2,0),9}, {(1,-2,0),-1}, {\
(-2,-1,0),-18}, {(-1,-1,0),27}, {(0,-1,0),-9}, {(-2,0,0),9}, {(-1,\
0,0),-9}, {(-2,1,0),-1}, {(1,1,0),1};

Tx26 = {(-2,-2,-2),56}, {(-1,-2,-2),-140}, {(0,-2,-2),120}, {(1,-2\
,-2),-40}, {(2,-2,-2),4}, {(-2,-1,-2),-105}, {(-1,-1,-2),240}, {(0\
,-1,-2),-180}, {(1,-1,-2),48}, {(2,-1,-2),-3}, {(-2,0,-2),60}, {(-\
1,0,-2),-120}, {(0,0,-2),72}, {(1,0,-2),-12}, {(-2,1,-2),-10}, {(-\
1,1,-2),16}, {(0,1,-2),-6}, {(-2,-2,-1),-70}, {(-1,-2,-1),160}, {(\
0,-2,-1),-120}, {(1,-2,-1),32}, {(2,-2,-1),-2}, {(-2,-1,-1),120}, \
{(-1,-1,-1),-240}, {(0,-1,-1),144}, {(1,-1,-1),-24}, {(-2,0,-1),-6\
0}, {(-1,0,-1),96}, {(0,0,-1),-36}, {(-2,1,-1),8}, {(-1,1,-1),-8},\
 {(-2,-2,0),20}, {(-1,-2,0),-40}, {(0,-2,0),24}, {(1,-2,0),-4}, {(\
-2,-1,0),-30}, {(-1,-1,0),48}, {(0,-1,0),-18}, {(-2,0,0),12}, {(-1\
,0,0),-12}, {(-2,1,0),-1}, {(2,1,0),1};

Tx27 = {(-2,-2,-2),1}, {(-2,-1,-2),-4}, {(-2,0,-2),6}, {(-2,1,-2),\
-4}, {(-2,2,-2),1}, {(-2,-2,-1),-2}, {(-2,-1,-1),8}, {(-2,0,-1),-1\
2}, {(-2,1,-1),8}, {(-2,2,-1),-2}, {(-2,-2,0),1}, {(-2,-1,0),-4}, \
{(-2,0,0),6}, {(-2,1,0),-4}, {(-2,2,0),1};
```

```
Tx28 = {(-2,-2,-2),6}, {(-1,-2,-2),-5}, {(-2,-1,-2),-20}, {(-1,-1,\
-2),16}, {(-2,0,-2),24}, {(-1,0,-2),-18}, {(-2,1,-2),-12}, {(-1,1,\
-2),8}, {(-2,2,-2),2}, {(-1,2,-2),-1}, {(-2,-2,-1),-10}, {(-1,-2,-\
1),8}, {(-2,-1,-1),32}, {(-1,-1,-1),-24}, {(-2,0,-1),-36}, {(-1,0,\
-1),24}, {(-2,1,-1),16}, {(-1,1,-1),-8}, {(-2,2,-1),-2}, {(-2,-2,0\
),4}, {(-1,-2,0),-3}, {(-2,-1,0),-12}, {(-1,-1,0),8}, {(-2,0,0),12\
}, {(-1,0,0),-6}, {(-2,1,0),-4}, {(-1,2,0),1};

Tx29 = {(-2,-2,-2),21}, {(-1,-2,-2),-30}, {(0,-2,-2),10}, {(-2,-1,\
-2),-60}, {(-1,-1,-2),80}, {(0,-1,-2),-24}, {(-2,0,-2),60}, {(-1,0\
,-2),-72}, {(0,0,-2),18}, {(-2,1,-2),-24}, {(-1,1,-2),24}, {(0,1,-\
2),-4}, {(-2,2,-2),3}, {(-1,2,-2),-2}, {(-2,-2,-1),-30}, {(-1,-2,-\
1),40}, {(0,-2,-1),-12}, {(-2,-1,-1),80}, {(-1,-1,-1),-96}, {(0,-1\
,-1),24}, {(-2,0,-1),-72}, {(-1,0,-1),72}, {(0,0,-1),-12}, {(-2,1,\
-1),24}, {(-1,1,-1),-16}, {(-2,2,-1),-2}, {(-2,-2,0),10}, {(-1,-2,\
0),-12}, {(0,-2,0),3}, {(-2,-1,0),-24}, {(-1,-1,0),24}, {(0,-1,0),\
-4}, {(-2,0,0),18}, {(-1,0,0),-12}, {(-2,1,0),-4}, {(0,2,0),1};

Tx30 = {(-2,-2,-2),56}, {(-1,-2,-2),-105}, {(0,-2,-2),60}, {(1,-2,\
-2),-10}, {(-2,-1,-2),-140}, {(-1,-1,-2),240}, {(0,-1,-2),-120}, {\
(1,-1,-2),16}, {(-2,0,-2),120}, {(-1,0,-2),-180}, {(0,0,-2),72}, {\
(1,0,-2),-6}, {(-2,1,-2),-40}, {(-1,1,-2),48}, {(0,1,-2),-12}, {(-\
2,2,-2),4}, {(-1,2,-2),-3}, {(-2,-2,-1),-70}, {(-1,-2,-1),120}, {(\
0,-2,-1),-60}, {(1,-2,-1),8}, {(-2,-1,-1),160}, {(-1,-1,-1),-240},\
 {(0,-1,-1),96}, {(1,-1,-1),-8}, {(-2,0,-1),-120}, {(-1,0,-1),144}\
, {(0,0,-1),-36}, {(-2,1,-1),32}, {(-1,1,-1),-24}, {(-2,2,-1),-2},\
 {(-2,-2,0),20}, {(-1,-2,0),-30}, {(0,-2,0),12}, {(1,-2,0),-1}, {(\
-2,-1,0),-40}, {(-1,-1,0),48}, {(0,-1,0),-12}, {(-2,0,0),24}, {(-1\
,0,0),-18}, {(-2,1,0),-4}, {(1,2,0),1};

Tx31 = {(-2,-2,-2),126}, {(-1,-2,-2),-280}, {(0,-2,-2),210}, {(1,-\
2,-2),-60}, {(2,-2,-2),5}, {(-2,-1,-2),-280}, {(-1,-1,-2),560}, {(\
0,-1,-2),-360}, {(1,-1,-2),80}, {(2,-1,-2),-4}, {(-2,0,-2),210}, {\
(-1,0,-2),-360}, {(0,0,-2),180}, {(1,0,-2),-24}, {(-2,1,-2),-60}, \
{(-1,1,-2),80}, {(0,1,-2),-24}, {(-2,2,-2),5}, {(-1,2,-2),-4}, {(-\
2,-2,-1),-140}, {(-1,-2,-1),280}, {(0,-2,-1),-180}, {(1,-2,-1),40}\
, {(2,-2,-1),-2}, {(-2,-1,-1),280}, {(-1,-1,-1),-480}, {(0,-1,-1),\
240}, {(1,-1,-1),-32}, {(-2,0,-1),-180}, {(-1,0,-1),240}, {(0,0,-1\
),-72}, {(-2,1,-1),40}, {(-1,1,-1),-32}, {(-2,2,-1),-2}, {(-2,-2,0\
),35}, {(-1,-2,0),-60}, {(0,-2,0),30}, {(1,-2,0),-4}, {(-2,-1,0),-\
```

```
60}, {(-1,-1,0),80}, {(0,-1,0),-24}, {(-2,0,0),30}, {(-1,0,0),-24}\
, {(-2,1,0),-4}, {(2,2,0),1};

Tx32 = {(-2,-2,-2),1}, {(-1,-2,-2),-3}, {(0,-2,-2),3}, {(1,-2,-2),\
-1}, {(-2,-2,-1),-3}, {(-1,-2,-1),9}, {(0,-2,-1),-9}, {(1,-2,-1),3\
}, {(-2,-2,0),3}, {(-1,-2,0),-9}, {(0,-2,0),9}, {(1,-2,0),-3}, {(-\
2,-2,1),-1}, {(-1,-2,1),3}, {(0,-2,1),-3}, {(1,-2,1),1};

Tx33 = {(-2,-2,-2),6}, {(-1,-2,-2),-20}, {(0,-2,-2),24}, {(1,-2,-2\
),-12}, {(2,-2,-2),2}, {(-2,-2,-1),-15}, {(-1,-2,-1),48}, {(0,-2,-\
1),-54}, {(1,-2,-1),24}, {(2,-2,-1),-3}, {(-2,-2,0),12}, {(-1,-2,0\
),-36}, {(0,-2,0),36}, {(1,-2,0),-12}, {(-2,-2,1),-3}, {(-1,-2,1),\
8}, {(0,-2,1),-6}, {(2,-2,1),1};

Tx34 = {(-2,-2,-2),1}, {(-1,-2,-2),-2}, {(0,-2,-2),1}, {(-2,-1,-2)\
,-1}, {(-1,-1,-2),2}, {(0,-1,-2),-1}, {(-2,-2,-1),-3}, {(-1,-2,-1)\
,6}, {(0,-2,-1),-3}, {(-2,-1,-1),3}, {(-1,-1,-1),-6}, {(0,-1,-1),3\
}, {(-2,-2,0),3}, {(-1,-2,0),-6}, {(0,-2,0),3}, {(-2,-1,0),-3}, {(\
-1,-1,0),6}, {(0,-1,0),-3}, {(-2,-2,1),-1}, {(-1,-2,1),2}, {(0,-2,\
1),-1}, {(-2,-1,1),1}, {(-1,-1,1),-2}, {(0,-1,1),1};

Tx35 = {(-2,-2,-2),6}, {(-1,-2,-2),-15}, {(0,-2,-2),12}, {(1,-2,-2\
),-3}, {(-2,-1,-2),-5}, {(-1,-1,-2),12}, {(0,-1,-2),-9}, {(1,-1,-2\
),2}, {(-2,-2,-1),-15}, {(-1,-2,-1),36}, {(0,-2,-1),-27}, {(1,-2,-\
1),6}, {(-2,-1,-1),12}, {(-1,-1,-1),-27}, {(0,-1,-1),18}, {(1,-1,-\
1),-3}, {(-2,-2,0),12}, {(-1,-2,0),-27}, {(0,-2,0),18}, {(1,-2,0),\
-3}, {(-2,-1,0),-9}, {(-1,-1,0),18}, {(0,-1,0),-9}, {(-2,-2,1),-3}\
, {(-1,-2,1),6}, {(0,-2,1),-3}, {(-2,-1,1),2}, {(-1,-1,1),-3}, {(1\
,-1,1),1};

Tx36 = {(-2,-2,-2),21}, {(-1,-2,-2),-60}, {(0,-2,-2),60}, {(1,-2,-\
2),-24}, {(2,-2,-2),3}, {(-2,-1,-2),-15}, {(-1,-1,-2),40}, {(0,-1,\
-2),-36}, {(1,-1,-2),12}, {(2,-1,-2),-1}, {(-2,-2,-1),-45}, {(-1,-\
2,-1),120}, {(0,-2,-1),-108}, {(1,-2,-1),36}, {(2,-2,-1),-3}, {(-2\
,-1,-1),30}, {(-1,-1,-1),-72}, {(0,-1,-1),54}, {(1,-1,-1),-12}, {(\
-2,-2,0),30}, {(-1,-2,0),-72}, {(0,-2,0),54}, {(1,-2,0),-12}, {(-2\
,-1,0),-18}, {(-1,-1,0),36}, {(0,-1,0),-18}, {(-2,-2,1),-6}, {(-1,\
-2,1),12}, {(0,-2,1),-6}, {(-2,-1,1),3}, {(-1,-1,1),-4}, {(2,-1,1)\
,1};
```

```
Tx37 = {(-2,-2,-2),1}, {(-1,-2,-2),-1}, {(-2,-1,-2),-2}, {(-1,-1,-\
2),2}, {(-2,0,-2),1}, {(-1,0,-2),-1}, {(-2,-2,-1),-3}, {(-1,-2,-1)\
,3}, {(-2,-1,-1),6}, {(-1,-1,-1),-6}, {(-2,0,-1),-3}, {(-1,0,-1),3\
}, {(-2,-2,0),3}, {(-1,-2,0),-3}, {(-2,-1,0),-6}, {(-1,-1,0),6}, {\
(-2,0,0),3}, {(-1,0,0),-3}, {(-2,-2,1),-1}, {(-1,-2,1),1}, {(-2,-1\
,1),2}, {(-1,-1,1),-2}, {(-2,0,1),-1}, {(-1,0,1),1};

Tx38 = {(-2,-2,-2),6}, {(-1,-2,-2),-10}, {(0,-2,-2),4}, {(-2,-1,-2\
),-10}, {(-1,-1,-2),16}, {(0,-1,-2),-6}, {(-2,0,-2),4}, {(-1,0,-2)\
,-6}, {(0,0,-2),2}, {(-2,-2,-1),-15}, {(-1,-2,-1),24}, {(0,-2,-1),\
-9}, {(-2,-1,-1),24}, {(-1,-1,-1),-36}, {(0,-1,-1),12}, {(-2,0,-1)\
,-9}, {(-1,0,-1),12}, {(0,0,-1),-3}, {(-2,-2,0),12}, {(-1,-2,0),-1\
8}, {(0,-2,0),6}, {(-2,-1,0),-18}, {(-1,-1,0),24}, {(0,-1,0),-6}, \
{(-2,0,0),6}, {(-1,0,0),-6}, {(-2,-2,1),-3}, {(-1,-2,1),4}, {(0,-2\
,1),-1}, {(-2,-1,1),4}, {(-1,-1,1),-4}, {(-2,0,1),-1}, {(0,0,1),1};

Tx39 = {(-2,-2,-2),21}, {(-1,-2,-2),-45}, {(0,-2,-2),30}, {(1,-2,-\
2),-6}, {(-2,-1,-2),-30}, {(-1,-1,-2),60}, {(0,-1,-2),-36}, {(1,-1\
,-2),6}, {(-2,0,-2),10}, {(-1,0,-2),-18}, {(0,0,-2),9}, {(1,0,-2),\
-1}, {(-2,-2,-1),-45}, {(-1,-2,-1),90}, {(0,-2,-1),-54}, {(1,-2,-1\
),9}, {(-2,-1,-1),60}, {(-1,-1,-1),-108}, {(0,-1,-1),54}, {(1,-1,-\
1),-6}, {(-2,0,-1),-18}, {(-1,0,-1),27}, {(0,0,-1),-9}, {(-2,-2,0)\
,30}, {(-1,-2,0),-54}, {(0,-2,0),27}, {(1,-2,0),-3}, {(-2,-1,0),-3\
6}, {(-1,-1,0),54}, {(0,-1,0),-18}, {(-2,0,0),9}, {(-1,0,0),-9}, {\
(-2,-2,1),-6}, {(-1,-2,1),9}, {(0,-2,1),-3}, {(-2,-1,1),6}, {(-1,-\
1,1),-6}, {(-2,0,1),-1}, {(1,0,1),1};

Tx40 = {(-2,-2,-2),56}, {(-1,-2,-2),-140}, {(0,-2,-2),120}, {(1,-2\
,-2),-40}, {(2,-2,-2),4}, {(-2,-1,-2),-70}, {(-1,-1,-2),160}, {(0,\
-1,-2),-120}, {(1,-1,-2),32}, {(2,-1,-2),-2}, {(-2,0,-2),20}, {(-1\
,0,-2),-40}, {(0,0,-2),24}, {(1,0,-2),-4}, {(-2,-2,-1),-105}, {(-1\
,-2,-1),240}, {(0,-2,-1),-180}, {(1,-2,-1),48}, {(2,-2,-1),-3}, {(\
-2,-1,-1),120}, {(-1,-1,-1),-240}, {(0,-1,-1),144}, {(1,-1,-1),-24\
}, {(-2,0,-1),-30}, {(-1,0,-1),48}, {(0,0,-1),-18}, {(-2,-2,0),60}\
, {(-1,-2,0),-120}, {(0,-2,0),72}, {(1,-2,0),-12}, {(-2,-1,0),-60}\
, {(-1,-1,0),96}, {(0,-1,0),-36}, {(-2,0,0),12}, {(-1,0,0),-12}, {\
(-2,-2,1),-10}, {(-1,-2,1),16}, {(0,-2,1),-6}, {(-2,-1,1),8}, {(-1\
,-1,1),-8}, {(-2,0,1),-1}, {(2,0,1),1};

Tx41 = {(-2,-2,-2),1}, {(-2,-1,-2),-3}, {(-2,0,-2),3}, {(-2,1,-2),\
```

```
-1}, {(-2,-2,-1),-3}, {(-2,-1,-1),9}, {(-2,0,-1),-9}, {(-2,1,-1),3\
}, {(-2,-2,0),3}, {(-2,-1,0),-9}, {(-2,0,0),9}, {(-2,1,0),-3}, {(-\
2,-2,1),-1}, {(-2,-1,1),3}, {(-2,0,1),-3}, {(-2,1,1),1};

Tx42 = {(-2,-2,-2),6}, {(-1,-2,-2),-5}, {(-2,-1,-2),-15}, {(-1,-1,\
-2),12}, {(-2,0,-2),12}, {(-1,0,-2),-9}, {(-2,1,-2),-3}, {(-1,1,-2\
),2}, {(-2,-2,-1),-15}, {(-1,-2,-1),12}, {(-2,-1,-1),36}, {(-1,-1,\
-1),-27}, {(-2,0,-1),-27}, {(-1,0,-1),18}, {(-2,1,-1),6}, {(-1,1,-\
1),-3}, {(-2,-2,0),12}, {(-1,-2,0),-9}, {(-2,-1,0),-27}, {(-1,-1,0\
),18}, {(-2,0,0),18}, {(-1,0,0),-9}, {(-2,1,0),-3}, {(-2,-2,1),-3}\
, {(-1,-2,1),2}, {(-2,-1,1),6}, {(-1,-1,1),-3}, {(-2,0,1),-3}, {(-\
1,1,1),1};

Tx43 = {(-2,-2,-2),21}, {(-1,-2,-2),-30}, {(0,-2,-2),10}, {(-2,-1,\
-2),-45}, {(-1,-1,-2),60}, {(0,-1,-2),-18}, {(-2,0,-2),30}, {(-1,0\
,-2),-36}, {(0,0,-2),9}, {(-2,1,-2),-6}, {(-1,1,-2),6}, {(0,1,-2),\
-1}, {(-2,-2,-1),-45}, {(-1,-2,-1),60}, {(0,-2,-1),-18}, {(-2,-1,-\
1),90}, {(-1,-1,-1),-108}, {(0,-1,-1),27}, {(-2,0,-1),-54}, {(-1,0\
,-1),54}, {(0,0,-1),-9}, {(-2,1,-1),9}, {(-1,1,-1),-6}, {(-2,-2,0)\
,30}, {(-1,-2,0),-36}, {(0,-2,0),9}, {(-2,-1,0),-54}, {(-1,-1,0),5\
4}, {(0,-1,0),-9}, {(-2,0,0),27}, {(-1,0,0),-18}, {(-2,1,0),-3}, {\
(-2,-2,1),-6}, {(-1,-2,1),6}, {(0,-2,1),-1}, {(-2,-1,1),9}, {(-1,-\
1,1),-6}, {(-2,0,1),-3}, {(0,1,1),1};

Tx44 = {(-2,-2,-2),56}, {(-1,-2,-2),-105}, {(0,-2,-2),60}, {(1,-2,\
-2),-10}, {(-2,-1,-2),-105}, {(-1,-1,-2),180}, {(0,-1,-2),-90}, {(\
1,-1,-2),12}, {(-2,0,-2),60}, {(-1,0,-2),-90}, {(0,0,-2),36}, {(1,\
0,-2),-3}, {(-2,1,-2),-10}, {(-1,1,-2),12}, {(0,1,-2),-3}, {(-2,-2\
,-1),-105}, {(-1,-2,-1),180}, {(0,-2,-1),-90}, {(1,-2,-1),12}, {(-\
2,-1,-1),180}, {(-1,-1,-1),-270}, {(0,-1,-1),108}, {(1,-1,-1),-9},\
 {(-2,0,-1),-90}, {(-1,0,-1),108}, {(0,0,-1),-27}, {(-2,1,-1),12},\
 {(-1,1,-1),-9}, {(-2,-2,0),60}, {(-1,-2,0),-90}, {(0,-2,0),36}, {\
(1,-2,0),-3}, {(-2,-1,0),-90}, {(-1,-1,0),108}, {(0,-1,0),-27}, {(\
-2,0,0),36}, {(-1,0,0),-27}, {(-2,1,0),-3}, {(-2,-2,1),-10}, {(-1,\
-2,1),12}, {(0,-2,1),-3}, {(-2,-1,1),12}, {(-1,-1,1),-9}, {(-2,0,1\
),-3}, {(1,1,1),1};

Tx45 = {(-2,-2,-2),126}, {(-1,-2,-2),-280}, {(0,-2,-2),210}, {(1,-\
2,-2),-60}, {(2,-2,-2),5}, {(-2,-1,-2),-210}, {(-1,-1,-2),420}, {(\
0,-1,-2),-270}, {(1,-1,-2),60}, {(2,-1,-2),-3}, {(-2,0,-2),105}, {\
```

```
(-1,0,-2),-180}, {(0,0,-2),90}, {(1,0,-2),-12}, {(-2,1,-2),-15}, {\
(-1,1,-2),20}, {(0,1,-2),-6}, {(-2,-2,-1),-210}, {(-1,-2,-1),420},\
 {(0,-2,-1),-270}, {(1,-2,-1),60}, {(2,-2,-1),-3}, {(-2,-1,-1),315\
}, {(-1,-1,-1),-540}, {(0,-1,-1),270}, {(1,-1,-1),-36}, {(-2,0,-1)\
,-135}, {(-1,0,-1),180}, {(0,0,-1),-54}, {(-2,1,-1),15}, {(-1,1,-1\
),-12}, {(-2,-2,0),105}, {(-1,-2,0),-180}, {(0,-2,0),90}, {(1,-2,0\
),-12}, {(-2,-1,0),-135}, {(-1,-1,0),180}, {(0,-1,0),-54}, {(-2,0,\
0),45}, {(-1,0,0),-36}, {(-2,1,0),-3}, {(-2,-2,1),-15}, {(-1,-2,1)\
,20}, {(0,-2,1),-6}, {(-2,-1,1),15}, {(-1,-1,1),-12}, {(-2,0,1),-3\
}, {(2,1,1),1};

Tx46 = {(-2,-2,-2),6}, {(-2,-1,-2),-20}, {(-2,0,-2),24}, {(-2,1,-2\
),-12}, {(-2,2,-2),2}, {(-2,-2,-1),-15}, {(-2,-1,-1),48}, {(-2,0,-\
1),-54}, {(-2,1,-1),24}, {(-2,2,-1),-3}, {(-2,-2,0),12}, {(-2,-1,0\
),-36}, {(-2,0,0),36}, {(-2,1,0),-12}, {(-2,-2,1),-3}, {(-2,-1,1),\
8}, {(-2,0,1),-6}, {(-2,2,1),1};

Tx47 = {(-2,-2,-2),21}, {(-1,-2,-2),-15}, {(-2,-1,-2),-60}, {(-1,-\
1,-2),40}, {(-2,0,-2),60}, {(-1,0,-2),-36}, {(-2,1,-2),-24}, {(-1,\
1,-2),12}, {(-2,2,-2),3}, {(-1,2,-2),-1}, {(-2,-2,-1),-45}, {(-1,-\
2,-1),30}, {(-2,-1,-1),120}, {(-1,-1,-1),-72}, {(-2,0,-1),-108}, {\
(-1,0,-1),54}, {(-2,1,-1),36}, {(-1,1,-1),-12}, {(-2,2,-1),-3}, {(\
-2,-2,0),30}, {(-1,-2,0),-18}, {(-2,-1,0),-72}, {(-1,-1,0),36}, {(\
-2,0,0),54}, {(-1,0,0),-18}, {(-2,1,0),-12}, {(-2,-2,1),-6}, {(-1,\
-2,1),3}, {(-2,-1,1),12}, {(-1,-1,1),-4}, {(-2,0,1),-6}, {(-1,2,1)\
,1};

Tx48 = {(-2,-2,-2),56}, {(-1,-2,-2),-70}, {(0,-2,-2),20}, {(-2,-1,\
-2),-140}, {(-1,-1,-2),160}, {(0,-1,-2),-40}, {(-2,0,-2),120}, {(-\
1,0,-2),-120}, {(0,0,-2),24}, {(-2,1,-2),-40}, {(-1,1,-2),32}, {(0\
,1,-2),-4}, {(-2,2,-2),4}, {(-1,2,-2),-2}, {(-2,-2,-1),-105}, {(-1\
,-2,-1),120}, {(0,-2,-1),-30}, {(-2,-1,-1),240}, {(-1,-1,-1),-240}\
, {(0,-1,-1),48}, {(-2,0,-1),-180}, {(-1,0,-1),144}, {(0,0,-1),-18\
}, {(-2,1,-1),48}, {(-1,1,-1),-24}, {(-2,2,-1),-3}, {(-2,-2,0),60}\
, {(-1,-2,0),-60}, {(0,-2,0),12}, {(-2,-1,0),-120}, {(-1,-1,0),96}\
, {(0,-1,0),-12}, {(-2,0,0),72}, {(-1,0,0),-36}, {(-2,1,0),-12}, {\
(-2,-2,1),-10}, {(-1,-2,1),8}, {(0,-2,1),-1}, {(-2,-1,1),16}, {(-1\
,-1,1),-8}, {(-2,0,1),-6}, {(0,2,1),1};

Tx49 = {(-2,-2,-2),126}, {(-1,-2,-2),-210}, {(0,-2,-2),105}, {(1,-\
```

```
2,-2),-15}, {(-2,-1,-2),-280}, {(-1,-1,-2),420}, {(0,-1,-2),-180},\
 {(1,-1,-2),20}, {(-2,0,-2),210}, {(-1,0,-2),-270}, {(0,0,-2),90},\
 {(1,0,-2),-6}, {(-2,1,-2),-60}, {(-1,1,-2),60}, {(0,1,-2),-12}, {\
(-2,2,-2),5}, {(-1,2,-2),-3}, {(-2,-2,-1),-210}, {(-1,-2,-1),315},\
 {(0,-2,-1),-135}, {(1,-2,-1),15}, {(-2,-1,-1),420}, {(-1,-1,-1),-\
540}, {(0,-1,-1),180}, {(1,-1,-1),-12}, {(-2,0,-1),-270}, {(-1,0,-\
1),270}, {(0,0,-1),-54}, {(-2,1,-1),60}, {(-1,1,-1),-36}, {(-2,2,-\
1),-3}, {(-2,-2,0),105}, {(-1,-2,0),-135}, {(0,-2,0),45}, {(1,-2,0\
),-3}, {(-2,-1,0),-180}, {(-1,-1,0),180}, {(0,-1,0),-36}, {(-2,0,0\
),90}, {(-1,0,0),-54}, {(-2,1,0),-12}, {(-2,-2,1),-15}, {(-1,-2,1)\
,15}, {(0,-2,1),-3}, {(-2,-1,1),20}, {(-1,-1,1),-12}, {(-2,0,1),-6\
}, {(1,2,1),1};

Tx50 = {(-2,-2,-2),252}, {(-1,-2,-2),-504}, {(0,-2,-2),336}, {(1,-\
2,-2),-84}, {(2,-2,-2),6}, {(-2,-1,-2),-504}, {(-1,-1,-2),896}, {(\
0,-1,-2),-504}, {(1,-1,-2),96}, {(2,-1,-2),-4}, {(-2,0,-2),336}, {\
(-1,0,-2),-504}, {(0,0,-2),216}, {(1,0,-2),-24}, {(-2,1,-2),-84}, \
{(-1,1,-2),96}, {(0,1,-2),-24}, {(-2,2,-2),6}, {(-1,2,-2),-4}, {(-\
2,-2,-1),-378}, {(-1,-2,-1),672}, {(0,-2,-1),-378}, {(1,-2,-1),72}\
, {(2,-2,-1),-3}, {(-2,-1,-1),672}, {(-1,-1,-1),-1008}, {(0,-1,-1)\
,432}, {(1,-1,-1),-48}, {(-2,0,-1),-378}, {(-1,0,-1),432}, {(0,0,-\
1),-108}, {(-2,1,-1),72}, {(-1,1,-1),-48}, {(-2,2,-1),-3}, {(-2,-2\
,0),168}, {(-1,-2,0),-252}, {(0,-2,0),108}, {(1,-2,0),-12}, {(-2,-\
1,0),-252}, {(-1,-1,0),288}, {(0,-1,0),-72}, {(-2,0,0),108}, {(-1,\
0,0),-72}, {(-2,1,0),-12}, {(-2,-2,1),-21}, {(-1,-2,1),24}, {(0,-2\
,1),-6}, {(-2,-1,1),24}, {(-1,-1,1),-16}, {(-2,0,1),-6}, {(2,2,1),\
1};

Tx51 = {(-2,-2,-2),1}, {(-1,-2,-2),-2}, {(0,-2,-2),1}, {(-2,-2,-1)\
,-4}, {(-1,-2,-1),8}, {(0,-2,-1),-4}, {(-2,-2,0),6}, {(-1,-2,0),-1\
2}, {(0,-2,0),6}, {(-2,-2,1),-4}, {(-1,-2,1),8}, {(0,-2,1),-4}, {(\
-2,-2,2),1}, {(-1,-2,2),-2}, {(0,-2,2),1};

Tx52 = {(-2,-2,-2),6}, {(-1,-2,-2),-15}, {(0,-2,-2),12}, {(1,-2,-2\
),-3}, {(-2,-2,-1),-20}, {(-1,-2,-1),48}, {(0,-2,-1),-36}, {(1,-2,\
-1),8}, {(-2,-2,0),24}, {(-1,-2,0),-54}, {(0,-2,0),36}, {(1,-2,0),\
-6}, {(-2,-2,1),-12}, {(-1,-2,1),24}, {(0,-2,1),-12}, {(-2,-2,2),2\
}, {(-1,-2,2),-3}, {(1,-2,2),1};

Tx53 = {(-2,-2,-2),21}, {(-1,-2,-2),-60}, {(0,-2,-2),60}, {(1,-2,-\
```

```
2),-24}, {(2,-2,-2),3}, {(-2,-2,-1),-60}, {(-1,-2,-1),160}, {(0,-2\
,-1),-144}, {(1,-2,-1),48}, {(2,-2,-1),-4}, {(-2,-2,0),60}, {(-1,-\
2,0),-144}, {(0,-2,0),108}, {(1,-2,0),-24}, {(-2,-2,1),-24}, {(-1,\
-2,1),48}, {(0,-2,1),-24}, {(-2,-2,2),3}, {(-1,-2,2),-4}, {(2,-2,2\
),1};

Tx54 = {(-2,-2,-2),1}, {(-1,-2,-2),-1}, {(-2,-1,-2),-1}, {(-1,-1,-\
2),1}, {(-2,-2,-1),-4}, {(-1,-2,-1),4}, {(-2,-1,-1),4}, {(-1,-1,-1\
),-4}, {(-2,-2,0),6}, {(-1,-2,0),-6}, {(-2,-1,0),-6}, {(-1,-1,0),6\
}, {(-2,-2,1),-4}, {(-1,-2,1),4}, {(-2,-1,1),4}, {(-1,-1,1),-4}, {\
(-2,-2,2),1}, {(-1,-2,2),-1}, {(-2,-1,2),-1}, {(-1,-1,2),1};

Tx55 = {(-2,-2,-2),6}, {(-1,-2,-2),-10}, {(0,-2,-2),4}, {(-2,-1,-2\
),-5}, {(-1,-1,-2),8}, {(0,-1,-2),-3}, {(-2,-2,-1),-20}, {(-1,-2,-\
1),32}, {(0,-2,-1),-12}, {(-2,-1,-1),16}, {(-1,-1,-1),-24}, {(0,-1\
,-1),8}, {(-2,-2,0),24}, {(-1,-2,0),-36}, {(0,-2,0),12}, {(-2,-1,0\
),-18}, {(-1,-1,0),24}, {(0,-1,0),-6}, {(-2,-2,1),-12}, {(-1,-2,1)\
,16}, {(0,-2,1),-4}, {(-2,-1,1),8}, {(-1,-1,1),-8}, {(-2,-2,2),2},\
 {(-1,-2,2),-2}, {(-2,-1,2),-1}, {(0,-1,2),1};

Tx56 = {(-2,-2,-2),21}, {(-1,-2,-2),-45}, {(0,-2,-2),30}, {(1,-2,-\
2),-6}, {(-2,-1,-2),-15}, {(-1,-1,-2),30}, {(0,-1,-2),-18}, {(1,-1\
,-2),3}, {(-2,-2,-1),-60}, {(-1,-2,-1),120}, {(0,-2,-1),-72}, {(1,\
-2,-1),12}, {(-2,-1,-1),40}, {(-1,-1,-1),-72}, {(0,-1,-1),36}, {(1\
,-1,-1),-4}, {(-2,-2,0),60}, {(-1,-2,0),-108}, {(0,-2,0),54}, {(1,\
-2,0),-6}, {(-2,-1,0),-36}, {(-1,-1,0),54}, {(0,-1,0),-18}, {(-2,-\
2,1),-24}, {(-1,-2,1),36}, {(0,-2,1),-12}, {(-2,-1,1),12}, {(-1,-1\
,1),-12}, {(-2,-2,2),3}, {(-1,-2,2),-3}, {(-2,-1,2),-1}, {(1,-1,2)\
,1};

Tx57 = {(-2,-2,-2),56}, {(-1,-2,-2),-140}, {(0,-2,-2),120}, {(1,-2\
,-2),-40}, {(2,-2,-2),4}, {(-2,-1,-2),-35}, {(-1,-1,-2),80}, {(0,-\
1,-2),-60}, {(1,-1,-2),16}, {(2,-1,-2),-1}, {(-2,-2,-1),-140}, {(-\
1,-2,-1),320}, {(0,-2,-1),-240}, {(1,-2,-1),64}, {(2,-2,-1),-4}, {\
(-2,-1,-1),80}, {(-1,-1,-1),-160}, {(0,-1,-1),96}, {(1,-1,-1),-16}\
, {(-2,-2,0),120}, {(-1,-2,0),-240}, {(0,-2,0),144}, {(1,-2,0),-24\
}, {(-2,-1,0),-60}, {(-1,-1,0),96}, {(0,-1,0),-36}, {(-2,-2,1),-40\
}, {(-1,-2,1),64}, {(0,-2,1),-24}, {(-2,-1,1),16}, {(-1,-1,1),-16}\
, {(-2,-2,2),4}, {(-1,-2,2),-4}, {(-2,-1,2),-1}, {(2,-1,2),1};
```

```
Tx58 = {(-2,-2,-2),1}, {(-2,-1,-2),-2}, {(-2,0,-2),1}, {(-2,-2,-1)\
,-4}, {(-2,-1,-1),8}, {(-2,0,-1),-4}, {(-2,-2,0),6}, {(-2,-1,0),-1\
2}, {(-2,0,0),6}, {(-2,-2,1),-4}, {(-2,-1,1),8}, {(-2,0,1),-4}, {(\
-2,-2,2),1}, {(-2,-1,2),-2}, {(-2,0,2),1};

Tx59 = {(-2,-2,-2),6}, {(-1,-2,-2),-5}, {(-2,-1,-2),-10}, {(-1,-1,\
-2),8}, {(-2,0,-2),4}, {(-1,0,-2),-3}, {(-2,-2,-1),-20}, {(-1,-2,-\
1),16}, {(-2,-1,-1),32}, {(-1,-1,-1),-24}, {(-2,0,-1),-12}, {(-1,0\
,-1),8}, {(-2,-2,0),24}, {(-1,-2,0),-18}, {(-2,-1,0),-36}, {(-1,-1\
,0),24}, {(-2,0,0),12}, {(-1,0,0),-6}, {(-2,-2,1),-12}, {(-1,-2,1)\
,8}, {(-2,-1,1),16}, {(-1,-1,1),-8}, {(-2,0,1),-4}, {(-2,-2,2),2},\
 {(-1,-2,2),-1}, {(-2,-1,2),-2}, {(-1,0,2),1};

Tx60 = {(-2,-2,-2),21}, {(-1,-2,-2),-30}, {(0,-2,-2),10}, {(-2,-1,\
-2),-30}, {(-1,-1,-2),40}, {(0,-1,-2),-12}, {(-2,0,-2),10}, {(-1,0\
,-2),-12}, {(0,0,-2),3}, {(-2,-2,-1),-60}, {(-1,-2,-1),80}, {(0,-2\
,-1),-24}, {(-2,-1,-1),80}, {(-1,-1,-1),-96}, {(0,-1,-1),24}, {(-2\
,0,-1),-24}, {(-1,0,-1),24}, {(0,0,-1),-4}, {(-2,-2,0),60}, {(-1,-\
2,0),-72}, {(0,-2,0),18}, {(-2,-1,0),-72}, {(-1,-1,0),72}, {(0,-1,\
0),-12}, {(-2,0,0),18}, {(-1,0,0),-12}, {(-2,-2,1),-24}, {(-1,-2,1\
),24}, {(0,-2,1),-4}, {(-2,-1,1),24}, {(-1,-1,1),-16}, {(-2,0,1),-\
4}, {(-2,-2,2),3}, {(-1,-2,2),-2}, {(-2,-1,2),-2}, {(0,0,2),1};

Tx61 = {(-2,-2,-2),56}, {(-1,-2,-2),-105}, {(0,-2,-2),60}, {(1,-2,\
-2),-10}, {(-2,-1,-2),-70}, {(-1,-1,-2),120}, {(0,-1,-2),-60}, {(1\
,-1,-2),8}, {(-2,0,-2),20}, {(-1,0,-2),-30}, {(0,0,-2),12}, {(1,0,\
-2),-1}, {(-2,-2,-1),-140}, {(-1,-2,-1),240}, {(0,-2,-1),-120}, {(\
1,-2,-1),16}, {(-2,-1,-1),160}, {(-1,-1,-1),-240}, {(0,-1,-1),96},\
 {(1,-1,-1),-8}, {(-2,0,-1),-40}, {(-1,0,-1),48}, {(0,0,-1),-12}, \
{(-2,-2,0),120}, {(-1,-2,0),-180}, {(0,-2,0),72}, {(1,-2,0),-6}, {\
(-2,-1,0),-120}, {(-1,-1,0),144}, {(0,-1,0),-36}, {(-2,0,0),24}, {\
(-1,0,0),-18}, {(-2,-2,1),-40}, {(-1,-2,1),48}, {(0,-2,1),-12}, {(\
-2,-1,1),32}, {(-1,-1,1),-24}, {(-2,0,1),-4}, {(-2,-2,2),4}, {(-1,\
-2,2),-3}, {(-2,-1,2),-2}, {(1,0,2),1};

Tx62 = {(-2,-2,-2),126}, {(-1,-2,-2),-280}, {(0,-2,-2),210}, {(1,-\
2,-2),-60}, {(2,-2,-2),5}, {(-2,-1,-2),-140}, {(-1,-1,-2),280}, {(\
0,-1,-2),-180}, {(1,-1,-2),40}, {(2,-1,-2),-2}, {(-2,0,-2),35}, {(\
-1,0,-2),-60}, {(0,0,-2),30}, {(1,0,-2),-4}, {(-2,-2,-1),-280}, {(\
-1,-2,-1),560}, {(0,-2,-1),-360}, {(1,-2,-1),80}, {(2,-2,-1),-4}, \
```

```
{(-2,-1,-1),280}, {(-1,-1,-1),-480}, {(0,-1,-1),240}, {(1,-1,-1),-\
32}, {(-2,0,-1),-60}, {(-1,0,-1),80}, {(0,0,-1),-24}, {(-2,-2,0),2\
10}, {(-1,-2,0),-360}, {(0,-2,0),180}, {(1,-2,0),-24}, {(-2,-1,0),\
-180}, {(-1,-1,0),240}, {(0,-1,0),-72}, {(-2,0,0),30}, {(-1,0,0),-\
24}, {(-2,-2,1),-60}, {(-1,-2,1),80}, {(0,-2,1),-24}, {(-2,-1,1),4\
0}, {(-1,-1,1),-32}, {(-2,0,1),-4}, {(-2,-2,2),5}, {(-1,-2,2),-4},\
 {(-2,-1,2),-2}, {(2,0,2),1};

Tx63 = {(-2,-2,-2),6}, {(-2,-1,-2),-15}, {(-2,0,-2),12}, {(-2,1,-2\
),-3}, {(-2,-2,-1),-20}, {(-2,-1,-1),48}, {(-2,0,-1),-36}, {(-2,1,\
-1),8}, {(-2,-2,0),24}, {(-2,-1,0),-54}, {(-2,0,0),36}, {(-2,1,0),\
-6}, {(-2,-2,1),-12}, {(-2,-1,1),24}, {(-2,0,1),-12}, {(-2,-2,2),2\
}, {(-2,-1,2),-3}, {(-2,1,2),1};

Tx64 = {(-2,-2,-2),21}, {(-1,-2,-2),-15}, {(-2,-1,-2),-45}, {(-1,-\
1,-2),30}, {(-2,0,-2),30}, {(-1,0,-2),-18}, {(-2,1,-2),-6}, {(-1,1\
,-2),3}, {(-2,-2,-1),-60}, {(-1,-2,-1),40}, {(-2,-1,-1),120}, {(-1\
,-1,-1),-72}, {(-2,0,-1),-72}, {(-1,0,-1),36}, {(-2,1,-1),12}, {(-\
1,1,-1),-4}, {(-2,-2,0),60}, {(-1,-2,0),-36}, {(-2,-1,0),-108}, {(\
-1,-1,0),54}, {(-2,0,0),54}, {(-1,0,0),-18}, {(-2,1,0),-6}, {(-2,-\
2,1),-24}, {(-1,-2,1),12}, {(-2,-1,1),36}, {(-1,-1,1),-12}, {(-2,0\
,1),-12}, {(-2,-2,2),3}, {(-1,-2,2),-1}, {(-2,-1,2),-3}, {(-1,1,2)\
,1};

Tx65 = {(-2,-2,-2),56}, {(-1,-2,-2),-70}, {(0,-2,-2),20}, {(-2,-1,\
-2),-105}, {(-1,-1,-2),120}, {(0,-1,-2),-30}, {(-2,0,-2),60}, {(-1\
,0,-2),-60}, {(0,0,-2),12}, {(-2,1,-2),-10}, {(-1,1,-2),8}, {(0,1,\
-2),-1}, {(-2,-2,-1),-140}, {(-1,-2,-1),160}, {(0,-2,-1),-40}, {(-\
2,-1,-1),240}, {(-1,-1,-1),-240}, {(0,-1,-1),48}, {(-2,0,-1),-120}\
, {(-1,0,-1),96}, {(0,0,-1),-12}, {(-2,1,-1),16}, {(-1,1,-1),-8}, \
{(-2,-2,0),120}, {(-1,-2,0),-120}, {(0,-2,0),24}, {(-2,-1,0),-180}\
, {(-1,-1,0),144}, {(0,-1,0),-18}, {(-2,0,0),72}, {(-1,0,0),-36}, \
{(-2,1,0),-6}, {(-2,-2,1),-40}, {(-1,-2,1),32}, {(0,-2,1),-4}, {(-\
2,-1,1),48}, {(-1,-1,1),-24}, {(-2,0,1),-12}, {(-2,-2,2),4}, {(-1,\
-2,2),-2}, {(-2,-1,2),-3}, {(0,1,2),1};

Tx66 = {(-2,-2,-2),126}, {(-1,-2,-2),-210}, {(0,-2,-2),105}, {(1,-\
2,-2),-15}, {(-2,-1,-2),-210}, {(-1,-1,-2),315}, {(0,-1,-2),-135},\
 {(1,-1,-2),15}, {(-2,0,-2),105}, {(-1,0,-2),-135}, {(0,0,-2),45},\
 {(1,0,-2),-3}, {(-2,1,-2),-15}, {(-1,1,-2),15}, {(0,1,-2),-3}, {(\
```

```
-2,-2,-1),-280}, {(-1,-2,-1),420}, {(0,-2,-1),-180}, {(1,-2,-1),20\
}, {(-2,-1,-1),420}, {(-1,-1,-1),-540}, {(0,-1,-1),180}, {(1,-1,-1\
),-12}, {(-2,0,-1),-180}, {(-1,0,-1),180}, {(0,0,-1),-36}, {(-2,1,\
-1),20}, {(-1,1,-1),-12}, {(-2,-2,0),210}, {(-1,-2,0),-270}, {(0,-\
2,0),90}, {(1,-2,0),-6}, {(-2,-1,0),-270}, {(-1,-1,0),270}, {(0,-1\
,0),-54}, {(-2,0,0),90}, {(-1,0,0),-54}, {(-2,1,0),-6}, {(-2,-2,1)\
,-60}, {(-1,-2,1),60}, {(0,-2,1),-12}, {(-2,-1,1),60}, {(-1,-1,1),\
-36}, {(-2,0,1),-12}, {(-2,-2,2),5}, {(-1,-2,2),-3}, {(-2,-1,2),-3\
}, {(1,1,2),1};

Tx67 = {(-2,-2,-2),252}, {(-1,-2,-2),-504}, {(0,-2,-2),336}, {(1,-\
2,-2),-84}, {(2,-2,-2),6}, {(-2,-1,-2),-378}, {(-1,-1,-2),672}, {(\
0,-1,-2),-378}, {(1,-1,-2),72}, {(2,-1,-2),-3}, {(-2,0,-2),168}, {\
(-1,0,-2),-252}, {(0,0,-2),108}, {(1,0,-2),-12}, {(-2,1,-2),-21}, \
{(-1,1,-2),24}, {(0,1,-2),-6}, {(-2,-2,-1),-504}, {(-1,-2,-1),896}\
, {(0,-2,-1),-504}, {(1,-2,-1),96}, {(2,-2,-1),-4}, {(-2,-1,-1),67\
2}, {(-1,-1,-1),-1008}, {(0,-1,-1),432}, {(1,-1,-1),-48}, {(-2,0,-\
1),-252}, {(-1,0,-1),288}, {(0,0,-1),-72}, {(-2,1,-1),24}, {(-1,1,\
-1),-16}, {(-2,-2,0),336}, {(-1,-2,0),-504}, {(0,-2,0),216}, {(1,-\
2,0),-24}, {(-2,-1,0),-378}, {(-1,-1,0),432}, {(0,-1,0),-108}, {(-\
2,0,0),108}, {(-1,0,0),-72}, {(-2,1,0),-6}, {(-2,-2,1),-84}, {(-1,\
-2,1),96}, {(0,-2,1),-24}, {(-2,-1,1),72}, {(-1,-1,1),-48}, {(-2,0\
,1),-12}, {(-2,-2,2),6}, {(-1,-2,2),-4}, {(-2,-1,2),-3}, {(2,1,2),\
1};

Tx68 = {(-2,-2,-2),21}, {(-2,-1,-2),-60}, {(-2,0,-2),60}, {(-2,1,-\
2),-24}, {(-2,2,-2),3}, {(-2,-2,-1),-60}, {(-2,-1,-1),160}, {(-2,0\
,-1),-144}, {(-2,1,-1),48}, {(-2,2,-1),-4}, {(-2,-2,0),60}, {(-2,-\
1,0),-144}, {(-2,0,0),108}, {(-2,1,0),-24}, {(-2,-2,1),-24}, {(-2,\
-1,1),48}, {(-2,0,1),-24}, {(-2,-2,2),3}, {(-2,-1,2),-4}, {(-2,2,2\
),1};

Tx69 = {(-2,-2,-2),56}, {(-1,-2,-2),-35}, {(-2,-1,-2),-140}, {(-1,\
-1,-2),80}, {(-2,0,-2),120}, {(-1,0,-2),-60}, {(-2,1,-2),-40}, {(-\
1,1,-2),16}, {(-2,2,-2),4}, {(-1,2,-2),-1}, {(-2,-2,-1),-140}, {(-\
1,-2,-1),80}, {(-2,-1,-1),320}, {(-1,-1,-1),-160}, {(-2,0,-1),-240\
}, {(-1,0,-1),96}, {(-2,1,-1),64}, {(-1,1,-1),-16}, {(-2,2,-1),-4}\
, {(-2,-2,0),120}, {(-1,-2,0),-60}, {(-2,-1,0),-240}, {(-1,-1,0),9\
6}, {(-2,0,0),144}, {(-1,0,0),-36}, {(-2,1,0),-24}, {(-2,-2,1),-40\
}, {(-1,-2,1),16}, {(-2,-1,1),64}, {(-1,-1,1),-16}, {(-2,0,1),-24}\
```

```
, {(-2,-2,2),4}, {(-1,-2,2),-1}, {(-2,-1,2),-4}, {(-1,2,2),1};

Tx70 = {(-2,-2,-2),126}, {(-1,-2,-2),-140}, {(0,-2,-2),35}, {(-2,-\
1,-2),-280}, {(-1,-1,-2),280}, {(0,-1,-2),-60}, {(-2,0,-2),210}, {\
(-1,0,-2),-180}, {(0,0,-2),30}, {(-2,1,-2),-60}, {(-1,1,-2),40}, {\
(0,1,-2),-4}, {(-2,2,-2),5}, {(-1,2,-2),-2}, {(-2,-2,-1),-280}, {(\
-1,-2,-1),280}, {(0,-2,-1),-60}, {(-2,-1,-1),560}, {(-1,-1,-1),-48\
0}, {(0,-1,-1),80}, {(-2,0,-1),-360}, {(-1,0,-1),240}, {(0,0,-1),-\
24}, {(-2,1,-1),80}, {(-1,1,-1),-32}, {(-2,2,-1),-4}, {(-2,-2,0),2\
10}, {(-1,-2,0),-180}, {(0,-2,0),30}, {(-2,-1,0),-360}, {(-1,-1,0)\
,240}, {(0,-1,0),-24}, {(-2,0,0),180}, {(-1,0,0),-72}, {(-2,1,0),-\
24}, {(-2,-2,1),-60}, {(-1,-2,1),40}, {(0,-2,1),-4}, {(-2,-1,1),80\
}, {(-1,-1,1),-32}, {(-2,0,1),-24}, {(-2,-2,2),5}, {(-1,-2,2),-2},\
 {(-2,-1,2),-4}, {(0,2,2),1};

Tx71 = {(-2,-2,-2),252}, {(-1,-2,-2),-378}, {(0,-2,-2),168}, {(1,-\
2,-2),-21}, {(-2,-1,-2),-504}, {(-1,-1,-2),672}, {(0,-1,-2),-252},\
 {(1,-1,-2),24}, {(-2,0,-2),336}, {(-1,0,-2),-378}, {(0,0,-2),108}\
, {(1,0,-2),-6}, {(-2,1,-2),-84}, {(-1,1,-2),72}, {(0,1,-2),-12}, \
{(-2,2,-2),6}, {(-1,2,-2),-3}, {(-2,-2,-1),-504}, {(-1,-2,-1),672}\
, {(0,-2,-1),-252}, {(1,-2,-1),24}, {(-2,-1,-1),896}, {(-1,-1,-1),\
-1008}, {(0,-1,-1),288}, {(1,-1,-1),-16}, {(-2,0,-1),-504}, {(-1,0\
,-1),432}, {(0,0,-1),-72}, {(-2,1,-1),96}, {(-1,1,-1),-48}, {(-2,2\
,-1),-4}, {(-2,-2,0),336}, {(-1,-2,0),-378}, {(0,-2,0),108}, {(1,-\
2,0),-6}, {(-2,-1,0),-504}, {(-1,-1,0),432}, {(0,-1,0),-72}, {(-2,\
0,0),216}, {(-1,0,0),-108}, {(-2,1,0),-24}, {(-2,-2,1),-84}, {(-1,\
-2,1),72}, {(0,-2,1),-12}, {(-2,-1,1),96}, {(-1,-1,1),-48}, {(-2,0\
,1),-24}, {(-2,-2,2),6}, {(-1,-2,2),-3}, {(-2,-1,2),-4}, {(1,2,2),\
1};

Tx72 = {(-2,-2,-2),462}, {(-1,-2,-2),-840}, {(0,-2,-2),504}, {(1,-\
2,-2),-112}, {(2,-2,-2),7}, {(-2,-1,-2),-840}, {(-1,-1,-2),1344}, \
{(0,-1,-2),-672}, {(1,-1,-2),112}, {(2,-1,-2),-4}, {(-2,0,-2),504}\
, {(-1,0,-2),-672}, {(0,0,-2),252}, {(1,0,-2),-24}, {(-2,1,-2),-11\
2}, {(-1,1,-2),112}, {(0,1,-2),-24}, {(-2,2,-2),7}, {(-1,2,-2),-4}\
, {(-2,-2,-1),-840}, {(-1,-2,-1),1344}, {(0,-2,-1),-672}, {(1,-2,-\
1),112}, {(2,-2,-1),-4}, {(-2,-1,-1),1344}, {(-1,-1,-1),-1792}, {(\
0,-1,-1),672}, {(1,-1,-1),-64}, {(-2,0,-1),-672}, {(-1,0,-1),672},\
 {(0,0,-1),-144}, {(-2,1,-1),112}, {(-1,1,-1),-64}, {(-2,2,-1),-4}\
, {(-2,-2,0),504}, {(-1,-2,0),-672}, {(0,-2,0),252}, {(1,-2,0),-24\
```

}, {(−2,−1,0),−672}, {(−1,−1,0),672}, {(0,−1,0),−144}, {(−2,0,0),2\
52}, {(−1,0,0),−144}, {(−2,1,0),−24}, {(−2,−2,1),−112}, {(−1,−2,1)\
,112}, {(0,−2,1),−24}, {(−2,−1,1),112}, {(−1,−1,1),−64}, {(−2,0,1)\
,−24}, {(−2,−2,2),7}, {(−1,−2,2),−4}, {(−2,−1,2),−4}, {(2,2,2),1};

Dy du/dy = Tfy +

ky1 Ty1 + ky2 Ty2 + ky3 Ty3 + ky4 Ty4 + ky5 Ty5 + ky6 Ty6 +

ky7 Ty7 + ky8 Ty8 + ky9 Ty9 + ky10 Ty10 + ky11 Ty11 +

ky12 Ty12 + ky13 Ty13 + ky14 Ty14 + ky15 Ty15 + ky16 Ty16 +

ky17 Ty17 + ky18 Ty18 + ky19 Ty19 + ky20 Ty20 + ky21 Ty21 +

ky22 Ty22 + ky23 Ty23 + ky24 Ty24 + ky25 Ty25 + ky26 Ty26 +

ky27 Ty27 + ky28 Ty28 + ky29 Ty29 + ky30 Ty30 + ky31 Ty31 +

ky32 Ty32 + ky33 Ty33 + ky34 Ty34 + ky35 Ty35 + ky36 Ty36 +

ky37 Ty37 + ky38 Ty38 + ky39 Ty39 + ky40 Ty40 + ky41 Ty41 +

ky42 Ty42 + ky43 Ty43 + ky44 Ty44 + ky45 Ty45 + ky46 Ty46 +

ky47 Ty47 + ky48 Ty48 + ky49 Ty49 + ky50 Ty50 + ky51 Ty51 +

ky52 Ty52 + ky53 Ty53 + ky54 Ty54 + ky55 Ty55 + ky56 Ty56 +

ky57 Ty57 + ky58 Ty58 + ky59 Ty59 + ky60 Ty60 + ky61 Ty61 +

ky62 Ty62 + ky63 Ty63 + ky64 Ty64 + ky65 Ty65 + ky66 Ty66 +

ky67 Ty67 + ky68 Ty68 + ky69 Ty69 + ky70 Ty70 + ky71 Ty71 +

ky72 Ty72, where

Tfy = {(−2,−2,−2),−5/4}, {(−1,−2,−2),5/3}, {(0,−2,−2),1/2}, {(1,−2\
,−2),−1}, {(2,−2,−2),1/6}, {(−2,−1,−2),5/2}, {(−1,−1,−2),−4}, {(1,\
−1,−2),4/3}, {(2,−1,−2),−1/6}, {(−2,0,−2),−2/3}, {(−1,0,−2),2}, {(\

```
0,0,-2),-1/2}, {(1,0,-2),-1/3}, {(-2,1,-2),-1}, {(-1,1,-2),2/3}, {\
(-2,2,-2),5/12}, {(-1,2,-2),-1/3}, {(-2,-2,-1),25/6}, {(-1,-2,-1),\
-22/3}, {(0,-2,-1),3}, {(-2,-1,-1),-28/3}, {(-1,-1,-1),16}, {(0,-1\
,-1),-6}, {(-2,0,-1),6}, {(-1,0,-1),-10}, {(0,0,-1),3}, {(-2,1,-1)\
,-2/3}, {(-1,1,-1),4/3}, {(-2,2,-1),-1/6}, {(-2,-2,0),-11/6}, {(-1\
,-2,0),3}, {(0,-2,0),-1}, {(-2,-1,0),4}, {(-1,-1,0),-6}, {(0,-1,0)\
,1}, {(-2,0,0),-5/2}, {(-1,0,0),3}, {(-2,1,0),1/3};

Ty1 = {(-2,-2,-2),1}, {(-1,-2,-2),-4}, {(0,-2,-2),6}, {(1,-2,-2),-\
4}, {(2,-2,-2),1}, {(-2,-1,-2),-2}, {(-1,-1,-2),8}, {(0,-1,-2),-12\
}, {(1,-1,-2),8}, {(2,-1,-2),-2}, {(-2,0,-2),1}, {(-1,0,-2),-4}, {\
(0,0,-2),6}, {(1,0,-2),-4}, {(2,0,-2),1};
```

...

Some of the output is eliminated, since the stencils $Tyi = Txi$ for $i = 1..72$.

...

```
Ty71 = {(-2,-2,-2),252}, {(-1,-2,-2),-378}, {(0,-2,-2),168}, {(1,-\
2,-2),-21}, {(-2,-1,-2),-504}, {(-1,-1,-2),672}, {(0,-1,-2),-252},\
 {(1,-1,-2),24}, {(-2,0,-2),336}, {(-1,0,-2),-378}, {(0,0,-2),108}\
, {(1,0,-2),-6}, {(-2,1,-2),-84}, {(-1,1,-2),72}, {(0,1,-2),-12}, \
{(-2,2,-2),6}, {(-1,2,-2),-3}, {(-2,-2,-1),-504}, {(-1,-2,-1),672}\
, {(0,-2,-1),-252}, {(1,-2,-1),24}, {(-2,-1,-1),896}, {(-1,-1,-1),\
-1008}, {(0,-1,-1),288}, {(1,-1,-1),-16}, {(-2,0,-1),-504}, {(-1,0\
,-1),432}, {(0,0,-1),-72}, {(-2,1,-1),96}, {(-1,1,-1),-48}, {(-2,2\
,-1),-4}, {(-2,-2,0),336}, {(-1,-2,0),-378}, {(0,-2,0),108}, {(1,-\
2,0),-6}, {(-2,-1,0),-504}, {(-1,-1,0),432}, {(0,-1,0),-72}, {(-2,\
0,0),216}, {(-1,0,0),-108}, {(-2,1,0),-24}, {(-2,-2,1),-84}, {(-1,\
-2,1),72}, {(0,-2,1),-12}, {(-2,-1,1),96}, {(-1,-1,1),-48}, {(-2,0\
,1),-24}, {(-2,-2,2),6}, {(-1,-2,2),-3}, {(-2,-1,2),-4}, {(1,2,2),\
1};

Ty72 = {(-2,-2,-2),462}, {(-1,-2,-2),-840}, {(0,-2,-2),504}, {(1,-\
2,-2),-112}, {(2,-2,-2),7}, {(-2,-1,-2),-840}, {(-1,-1,-2),1344}, \
{(0,-1,-2),-672}, {(1,-1,-2),112}, {(2,-1,-2),-4}, {(-2,0,-2),504}\
, {(-1,0,-2),-672}, {(0,0,-2),252}, {(1,0,-2),-24}, {(-2,1,-2),-11\
2}, {(-1,1,-2),112}, {(0,1,-2),-24}, {(-2,2,-2),7}, {(-1,2,-2),-4}\
, {(-2,-2,-1),-840}, {(-1,-2,-1),1344}, {(0,-2,-1),-672}, {(1,-2,-\
```

```
1),112}, {(2,-2,-1),-4}, {(-2,-1,-1),1344}, {(-1,-1,-1),-1792}, {(\
0,-1,-1),672}, {(1,-1,-1),-64}, {(-2,0,-1),-672}, {(-1,0,-1),672},\
 {(0,0,-1),-144}, {(-2,1,-1),112}, {(-1,1,-1),-64}, {(-2,2,-1),-4}\
, {(-2,-2,0),504}, {(-1,-2,0),-672}, {(0,-2,0),252}, {(1,-2,0),-24\
}, {(-2,-1,0),-672}, {(-1,-1,0),672}, {(0,-1,0),-144}, {(-2,0,0),2\
52}, {(-1,0,0),-144}, {(-2,1,0),-24}, {(-2,-2,1),-112}, {(-1,-2,1)\
,112}, {(0,-2,1),-24}, {(-2,-1,1),112}, {(-1,-1,1),-64}, {(-2,0,1)\
,-24}, {(-2,-2,2),7}, {(-1,-2,2),-4}, {(-2,-1,2),-4}, {(2,2,2),1};
```

Dz du/dz = Tfz +

kz1 Tz1 + kz2 Tz2 + kz3 Tz3 + kz4 Tz4 + kz5 Tz5 + kz6 Tz6 +

kz7 Tz7 + kz8 Tz8 + kz9 Tz9 + kz10 Tz10 + kz11 Tz11 +

kz12 Tz12 + kz13 Tz13 + kz14 Tz14 + kz15 Tz15 + kz16 Tz16 +

kz17 Tz17 + kz18 Tz18 + kz19 Tz19 + kz20 Tz20 + kz21 Tz21 +

kz22 Tz22 + kz23 Tz23 + kz24 Tz24 + kz25 Tz25 + kz26 Tz26 +

kz27 Tz27 + kz28 Tz28 + kz29 Tz29 + kz30 Tz30 + kz31 Tz31 +

kz32 Tz32 + kz33 Tz33 + kz34 Tz34 + kz35 Tz35 + kz36 Tz36 +

kz37 Tz37 + kz38 Tz38 + kz39 Tz39 + kz40 Tz40 + kz41 Tz41 +

kz42 Tz42 + kz43 Tz43 + kz44 Tz44 + kz45 Tz45 + kz46 Tz46 +

kz47 Tz47 + kz48 Tz48 + kz49 Tz49 + kz50 Tz50 + kz51 Tz51 +

kz52 Tz52 + kz53 Tz53 + kz54 Tz54 + kz55 Tz55 + kz56 Tz56 +

kz57 Tz57 + kz58 Tz58 + kz59 Tz59 + kz60 Tz60 + kz61 Tz61 +

kz62 Tz62 + kz63 Tz63 + kz64 Tz64 + kz65 Tz65 + kz66 Tz66 +

kz67 Tz67 + kz68 Tz68 + kz69 Tz69 + kz70 Tz70 + kz71 Tz71 +

kz72 Tz72, where

```
Tfz = {(-2,-2,-2),19/4}, {(-1,-2,-2),-20/3}, {(0,-2,-2),9/2}, {(1,\
-2,-2),-5/3}, {(2,-2,-2),1/6}, {(-2,-1,-2),-20/3}, {(-1,-1,-2),16/\
3}, {(0,-1,-2),-2}, {(1,-1,-2),2/3}, {(-2,0,-2),9/2}, {(-1,0,-2),-\
2}, {(-2,1,-2),-5/3}, {(-1,1,-2),2/3}, {(-2,2,-2),1/6}, {(-2,-2,-1\
),-25/3}, {(-1,-2,-1),26/3}, {(0,-2,-1),-5}, {(1,-2,-1),2}, {(2,-2\
,-1),-1/6}, {(-2,-1,-1),26/3}, {(-1,-1,-1),-2}, {(1,-1,-1),-2/3}, \
{(-2,0,-1),-5}, {(-2,1,-1),2}, {(-1,1,-1),-2/3}, {(-2,2,-1),-1/6},\
 {(-2,-2,0),17/3}, {(-1,-2,0),-3}, {(0,-2,0),1/2}, {(1,-2,0),-1/3}\
, {(-2,-1,0),-3}, {(-1,-1,0),-4}, {(0,-1,0),2}, {(-2,0,0),1/2}, {(\
-1,0,0),2}, {(-2,1,0),-1/3}, {(-2,-2,1),-8/3}, {(-1,-2,1),4/3}, {(\
-2,-1,1),4/3}, {(-1,-1,1),2/3}, {(-2,-2,2),7/12}, {(-1,-2,2),-1/3}\
, {(-2,-1,2),-1/3};
```

```
Tz1 = {(-2,-2,-2),1}, {(-1,-2,-2),-4}, {(0,-2,-2),6}, {(1,-2,-2),-\
4}, {(2,-2,-2),1}, {(-2,-1,-2),-2}, {(-1,-1,-2),8}, {(0,-1,-2),-12\
}, {(1,-1,-2),8}, {(2,-1,-2),-2}, {(-2,0,-2),1}, {(-1,0,-2),-4}, {\
(0,0,-2),6}, {(1,0,-2),-4}, {(2,0,-2),1};
```

...

Some of the output is eliminated, since the stencils $Tzi = Txi$ for $i = 1..72$.

...

```
Tz71 = {(-2,-2,-2),252}, {(-1,-2,-2),-378}, {(0,-2,-2),168}, {(1,-\
2,-2),-21}, {(-2,-1,-2),-504}, {(-1,-1,-2),672}, {(0,-1,-2),-252},\
 {(1,-1,-2),24}, {(-2,0,-2),336}, {(-1,0,-2),-378}, {(0,0,-2),108}\
, {(1,0,-2),-6}, {(-2,1,-2),-84}, {(-1,1,-2),72}, {(0,1,-2),-12}, \
{(-2,2,-2),6}, {(-1,2,-2),-3}, {(-2,-2,-1),-504}, {(-1,-2,-1),672}\
, {(0,-2,-1),-252}, {(1,-2,-1),24}, {(-2,-1,-1),896}, {(-1,-1,-1),\
-1008}, {(0,-1,-1),288}, {(1,-1,-1),-16}, {(-2,0,-1),-504}, {(-1,0\
,-1),432}, {(0,0,-1),-72}, {(-2,1,-1),96}, {(-1,1,-1),-48}, {(-2,2\
,-1),-4}, {(-2,-2,0),336}, {(-1,-2,0),-378}, {(0,-2,0),108}, {(1,-\
2,0),-6}, {(-2,-1,0),-504}, {(-1,-1,0),432}, {(0,-1,0),-72}, {(-2,\
0,0),216}, {(-1,0,0),-108}, {(-2,1,0),-24}, {(-2,-2,1),-84}, {(-1,\
-2,1),72}, {(0,-2,1),-12}, {(-2,-1,1),96}, {(-1,-1,1),-48}, {(-2,0\
,1),-24}, {(-2,-2,2),6}, {(-1,-2,2),-3}, {(-2,-1,2),-4}, {(1,2,2),\
1};
```

```
Tz72 = {(-2,-2,-2),462}, {(-1,-2,-2),-840}, {(0,-2,-2),504}, {(1,-\
2,-2),-112}, {(2,-2,-2),7}, {(-2,-1,-2),-840}, {(-1,-1,-2),1344}, \
{(0,-1,-2),-672}, {(1,-1,-2),112}, {(2,-1,-2),-4}, {(-2,0,-2),504}\
, {(-1,0,-2),-672}, {(0,0,-2),252}, {(1,0,-2),-24}, {(-2,1,-2),-11\
2}, {(-1,1,-2),112}, {(0,1,-2),-24}, {(-2,2,-2),7}, {(-1,2,-2),-4}\
, {(-2,-2,-1),-840}, {(-1,-2,-1),1344}, {(0,-2,-1),-672}, {(1,-2,-\
1),112}, {(2,-2,-1),-4}, {(-2,-1,-1),1344}, {(-1,-1,-1),-1792}, {(\
0,-1,-1),672}, {(1,-1,-1),-64}, {(-2,0,-1),-672}, {(-1,0,-1),672},\
 {(0,0,-1),-144}, {(-2,1,-1),112}, {(-1,1,-1),-64}, {(-2,2,-1),-4}\
, {(-2,-2,0),504}, {(-1,-2,0),-672}, {(0,-2,0),252}, {(1,-2,0),-24\
}, {(-2,-1,0),-672}, {(-1,-1,0),672}, {(0,-1,0),-144}, {(-2,0,0),2\
52}, {(-1,0,0),-144}, {(-2,1,0),-24}, {(-2,-2,1),-112}, {(-1,-2,1)\
,112}, {(0,-2,1),-24}, {(-2,-1,1),112}, {(-1,-1,1),-64}, {(-2,0,1)\
,-24}, {(-2,-2,2),7}, {(-1,-2,2),-4}, {(-2,-1,2),-4}, {(2,2,2),1};
```

Appendix 12 : The output for D_{200}, Order 2, on the grid $(-1..1)^3$, optimize=0

```
----- preparations -----

gridmin = -1, gridmax = 1

se1 = 2, se2 = 0, se3 = 0

order = 2

optimize = 0

check = 0

----- setup equations -----

Discretization of a derivative in 3D with order 2 :
2 times a derivation along the direction e1,
0 times a derivation along the direction e2,
0 times a derivation along the direction e3,
the highest derivative is 2

Establishing the equations for a consistent discretization.

Computing the local derivative error terms.

Equations for optimizing a pure derivative along e1

The equations are ready to be solved.

the time spent is 1.16 s.

----- solve equations -----

Solving the equations for consistent approximations

The solution is written to the file : D_200-O_2-G_-11-OP_0/soln.mu
```

the time spent is 0.68 s.

----- analyze solution ------

The constraints specified in the file generate-discretization,

gridmin = -1, gridmax = 1,

se1 = 2, se2 = 0, se3 = 0,

order = 2,

optimize = 0,

check = 0,

are satisfied by the following solution, which is written using the notation k... for a free coefficient and where T... represents a stencil which is given by a list of nodes with corresponding weights in the notation {(i,j,k),w} where the node is given by the indices i,j,k and the weight by w, and where Dx, Dy and Dz are the grid spacings in the three coordinate directions, in the following approximations :
(a line which ends with \ is continued on the next line)

```
Dx Dx d2u/dxdx = Tfxx +

kxx1 Txx1 + kxx2 Txx2 + kxx3 Txx3 + kxx4 Txx4 + kxx5 Txx5 +

kxx6 Txx6 + kxx7 Txx7 + kxx8 Txx8 + kxx9 Txx9 + kxx10 Txx10,

where

Tfxx = {(-1,-1,-1),-1}, {(0,-1,-1),2}, {(1,-1,-1),-1}, {(-1,0,-1),\
1}, {(0,0,-1),-2}, {(1,0,-1),1}, {(-1,-1,0),1}, {(0,-1,0),-2}, {(1\
,-1,0),1};

Txx1 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-\
2}, {(0,0,-1),4}, {(1,0,-1),-2}, {(-1,1,-1),1}, {(0,1,-1),-2}, {(1\
```

```
,1,-1),1};

Txx2 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-\
1}, {(0,0,-1),2}, {(1,0,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),2}, {(1\
,-1,0),-1}, {(-1,0,0),1}, {(0,0,0),-2}, {(1,0,0),1};

Txx3 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-2}, {(0,0,-1),2\
}, {(-1,1,-1),1}, {(0,1,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),1}, {(-\
1,0,0),2}, {(0,0,0),-2}, {(-1,1,0),-1}, {(0,1,0),1};

Txx4 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-\
6}, {(0,0,-1),8}, {(1,0,-1),-2}, {(-1,1,-1),2}, {(0,1,-1),-2}, {(-\
1,-1,0),-3}, {(0,-1,0),4}, {(1,-1,0),-1}, {(-1,0,0),4}, {(0,0,0),-\
4}, {(-1,1,0),-1}, {(1,1,0),1};

Txx5 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,-1,0),-\
2}, {(0,-1,0),4}, {(1,-1,0),-2}, {(-1,-1,1),1}, {(0,-1,1),-2}, {(1\
,-1,1),1};

Txx6 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-1}, {(0,0,-1),1\
}, {(-1,-1,0),-2}, {(0,-1,0),2}, {(-1,0,0),2}, {(0,0,0),-2}, {(-1,\
-1,1),1}, {(0,-1,1),-1}, {(-1,0,1),-1}, {(0,0,1),1};

Txx7 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-\
3}, {(0,0,-1),4}, {(1,0,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),8}, {(1\
,-1,0),-2}, {(-1,0,0),4}, {(0,0,0),-4}, {(-1,-1,1),2}, {(0,-1,1),-\
2}, {(-1,0,1),-1}, {(1,0,1),1};

Txx8 = {(-1,-1,-1),1}, {(-1,0,-1),-2}, {(-1,1,-1),1}, {(-1,-1,0),-\
2}, {(-1,0,0),4}, {(-1,1,0),-2}, {(-1,-1,1),1}, {(-1,0,1),-2}, {(-\
1,1,1),1};

Txx9 = {(-1,-1,-1),4}, {(0,-1,-1),-3}, {(-1,0,-1),-6}, {(0,0,-1),4\
}, {(-1,1,-1),2}, {(0,1,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),4}, {(-\
1,0,0),8}, {(0,0,0),-4}, {(-1,1,0),-2}, {(-1,-1,1),2}, {(0,-1,1),-\
1}, {(-1,0,1),-2}, {(0,1,1),1};

Txx10 = {(-1,-1,-1),10}, {(0,-1,-1),-12}, {(1,-1,-1),3}, {(-1,0,-1\
),-12}, {(0,0,-1),12}, {(1,0,-1),-2}, {(-1,1,-1),3}, {(0,1,-1),-2}\
, {(-1,-1,0),-12}, {(0,-1,0),12}, {(1,-1,0),-2}, {(-1,0,0),12}, {(\
```

0,0,0),−8}, {(−1,1,0),−2}, {(−1,−1,1),3}, {(0,−1,1),−2}, {(−1,0,1)\
,−2}, {(1,1,1),1};

Dx Dy d2u/dxdy = Tfxy +

kxy1 Txy1 + kxy2 Txy2 + kxy3 Txy3 + kxy4 Txy4 + kxy5 Txy5 +

kxy6 Txy6 + kxy7 Txy7 + kxy8 Txy8 + kxy9 Txy9 + kxy10 Txy10,

where

Tfxy = {(−1,−1,−1),−1}, {(0,−1,−1),3/2}, {(1,−1,−1),−1/2}, {(−1,0,\
−1),3/2}, {(0,0,−1),−2}, {(1,0,−1),1/2}, {(−1,1,−1),−1/2}, {(0,1,−\
1),1/2}, {(−1,−1,0),1}, {(0,−1,0),−1}, {(−1,0,0),−1}, {(0,0,0),1};

Txy1 = {(−1,−1,−1),1}, {(0,−1,−1),−2}, {(1,−1,−1),1}, {(−1,0,−1),−\
2}, {(0,0,−1),4}, {(1,0,−1),−2}, {(−1,1,−1),1}, {(0,1,−1),−2}, {(1\
,1,−1),1};

Txy2 = {(−1,−1,−1),1}, {(0,−1,−1),−2}, {(1,−1,−1),1}, {(−1,0,−1),−\
1}, {(0,0,−1),2}, {(1,0,−1),−1}, {(−1,−1,0),−1}, {(0,−1,0),2}, {(1\
,−1,0),−1}, {(−1,0,0),1}, {(0,0,0),−2}, {(1,0,0),1};

Txy3 = {(−1,−1,−1),1}, {(0,−1,−1),−1}, {(−1,0,−1),−2}, {(0,0,−1),2\
}, {(−1,1,−1),1}, {(0,1,−1),−1}, {(−1,−1,0),−1}, {(0,−1,0),1}, {(−\
1,0,0),2}, {(0,0,0),−2}, {(−1,1,0),−1}, {(0,1,0),1};

Txy4 = {(−1,−1,−1),4}, {(0,−1,−1),−6}, {(1,−1,−1),2}, {(−1,0,−1),−\
6}, {(0,0,−1),8}, {(1,0,−1),−2}, {(−1,1,−1),2}, {(0,1,−1),−2}, {(−\
1,−1,0),−3}, {(0,−1,0),4}, {(1,−1,0),−1}, {(−1,0,0),4}, {(0,0,0),−\
4}, {(−1,1,0),−1}, {(1,1,0),1};

Txy5 = {(−1,−1,−1),1}, {(0,−1,−1),−2}, {(1,−1,−1),1}, {(−1,−1,0),−\
2}, {(0,−1,0),4}, {(1,−1,0),−2}, {(−1,−1,1),1}, {(0,−1,1),−2}, {(1\
,−1,1),1};

Txy6 = {(−1,−1,−1),1}, {(0,−1,−1),−1}, {(−1,0,−1),−1}, {(0,0,−1),1\
}, {(−1,−1,0),−2}, {(0,−1,0),2}, {(−1,0,0),2}, {(0,0,0),−2}, {(−1,\
−1,1),1}, {(0,−1,1),−1}, {(−1,0,1),−1}, {(0,0,1),1};

```
Txy7 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-\
3}, {(0,0,-1),4}, {(1,0,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),8}, {(1\
,-1,0),-2}, {(-1,0,0),4}, {(0,0,0),-4}, {(-1,-1,1),2}, {(0,-1,1),-\
2}, {(-1,0,1),-1}, {(1,0,1),1};

Txy8 = {(-1,-1,-1),1}, {(-1,0,-1),-2}, {(-1,1,-1),1}, {(-1,-1,0),-\
2}, {(-1,0,0),4}, {(-1,1,0),-2}, {(-1,-1,1),1}, {(-1,0,1),-2}, {(-\
1,1,1),1};

Txy9 = {(-1,-1,-1),4}, {(0,-1,-1),-3}, {(-1,0,-1),-6}, {(0,0,-1),4\
}, {(-1,1,-1),2}, {(0,1,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),4}, {(-\
1,0,0),8}, {(0,0,0),-4}, {(-1,1,0),-2}, {(-1,-1,1),2}, {(0,-1,1),-\
1}, {(-1,0,1),-2}, {(0,1,1),1};

Txy10 = {(-1,-1,-1),10}, {(0,-1,-1),-12}, {(1,-1,-1),3}, {(-1,0,-1\
),-12}, {(0,0,-1),12}, {(1,0,-1),-2}, {(-1,1,-1),3}, {(0,1,-1),-2}\
, {(-1,-1,0),-12}, {(0,-1,0),12}, {(1,-1,0),-2}, {(-1,0,0),12}, {(\
0,0,0),-8}, {(-1,1,0),-2}, {(-1,-1,1),3}, {(0,-1,1),-2}, {(-1,0,1)\
,-2}, {(1,1,1),1};
```

Dy Dy d2u/dydy = Tfyy +

kyy1 Tyy1 + kyy2 Tyy2 + kyy3 Tyy3 + kyy4 Tyy4 + kyy5 Tyy5 +

kyy6 Tyy6 + kyy7 Tyy7 + kyy8 Tyy8 + kyy9 Tyy9 + kyy10 Tyy10,

where

```
Tfyy = {(-1,-1,-1),-1}, {(0,-1,-1),1}, {(-1,0,-1),2}, {(0,0,-1),-2\
}, {(-1,1,-1),-1}, {(0,1,-1),1}, {(-1,-1,0),1}, {(-1,0,0),-2}, {(-\
1,1,0),1};

Tyy1 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-\
2}, {(0,0,-1),4}, {(1,0,-1),-2}, {(-1,1,-1),1}, {(0,1,-1),-2}, {(1\
,1,-1),1};

Tyy2 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-\
1}, {(0,0,-1),2}, {(1,0,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),2}, {(1\
,-1,0),-1}, {(-1,0,0),1}, {(0,0,0),-2}, {(1,0,0),1};
```

```
Tyy3 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-2}, {(0,0,-1),2\
}, {(-1,1,-1),1}, {(0,1,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),1}, {(-\
1,0,0),2}, {(0,0,0),-2}, {(-1,1,0),-1}, {(0,1,0),1};

Tyy4 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-\
6}, {(0,0,-1),8}, {(1,0,-1),-2}, {(-1,1,-1),2}, {(0,1,-1),-2}, {(-\
1,-1,0),-3}, {(0,-1,0),4}, {(1,-1,0),-1}, {(-1,0,0),4}, {(0,0,0),-\
4}, {(-1,1,0),-1}, {(1,1,0),1};

Tyy5 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,-1,0),-\
2}, {(0,-1,0),4}, {(1,-1,0),-2}, {(-1,-1,1),1}, {(0,-1,1),-2}, {(1\
,-1,1),1};

Tyy6 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-1}, {(0,0,-1),1\
}, {(-1,-1,0),-2}, {(0,-1,0),2}, {(-1,0,0),2}, {(0,0,0),-2}, {(-1,\
-1,1),1}, {(0,-1,1),-1}, {(-1,0,1),-1}, {(0,0,1),1};

Tyy7 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-\
3}, {(0,0,-1),4}, {(1,0,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),8}, {(1\
,-1,0),-2}, {(-1,0,0),4}, {(0,0,0),-4}, {(-1,-1,1),2}, {(0,-1,1),-\
2}, {(-1,0,1),-1}, {(1,0,1),1};

Tyy8 = {(-1,-1,-1),1}, {(-1,0,-1),-2}, {(-1,1,-1),1}, {(-1,-1,0),-\
2}, {(-1,0,0),4}, {(-1,1,0),-2}, {(-1,-1,1),1}, {(-1,0,1),-2}, {(-\
1,1,1),1};

Tyy9 = {(-1,-1,-1),4}, {(0,-1,-1),-3}, {(-1,0,-1),-6}, {(0,0,-1),4\
}, {(-1,1,-1),2}, {(0,1,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),4}, {(-\
1,0,0),8}, {(0,0,0),-4}, {(-1,1,0),-2}, {(-1,-1,1),2}, {(0,-1,1),-\
1}, {(-1,0,1),-2}, {(0,1,1),1};

Tyy10 = {(-1,-1,-1),10}, {(0,-1,-1),-12}, {(1,-1,-1),3}, {(-1,0,-1\
),-12}, {(0,0,-1),12}, {(1,0,-1),-2}, {(-1,1,-1),3}, {(0,1,-1),-2}\
, {(-1,-1,0),-12}, {(0,-1,0),12}, {(1,-1,0),-2}, {(-1,0,0),12}, {(\
0,0,0),-8}, {(-1,1,0),-2}, {(-1,-1,1),3}, {(0,-1,1),-2}, {(-1,0,1)\
,-2}, {(1,1,1),1};

                    Dx Dz d2u/dxdz = Tfxz +

kxz1 Txz1 + kxz2 Txz2 + kxz3 Txz3 + kxz4 Txz4 + kxz5 Txz5 +
```

```
kxz6 Txz6 + kxz7 Txz7 + kxz8 Txz8 + kxz9 Txz9 + kxz10 Txz10,
```

where

```
Tfxz = {(-1,-1,-1),-1}, {(0,-1,-1),3/2}, {(1,-1,-1),-1/2}, {(-1,0,\
-1),1}, {(0,0,-1),-1}, {(-1,-1,0),3/2}, {(0,-1,0),-2}, {(1,-1,0),1\
/2}, {(-1,0,0),-1}, {(0,0,0),1}, {(-1,-1,1),-1/2}, {(0,-1,1),1/2};

Txz1 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-\
2}, {(0,0,-1),4}, {(1,0,-1),-2}, {(-1,1,-1),1}, {(0,1,-1),-2}, {(1\
,1,-1),1};

Txz2 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-\
1}, {(0,0,-1),2}, {(1,0,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),2}, {(1\
,-1,0),-1}, {(-1,0,0),1}, {(0,0,0),-2}, {(1,0,0),1};

Txz3 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-2}, {(0,0,-1),2\
}, {(-1,1,-1),1}, {(0,1,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),1}, {(-\
1,0,0),2}, {(0,0,0),-2}, {(-1,1,0),-1}, {(0,1,0),1};

Txz4 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-\
6}, {(0,0,-1),8}, {(1,0,-1),-2}, {(-1,1,-1),2}, {(0,1,-1),-2}, {(-\
1,-1,0),-3}, {(0,-1,0),4}, {(1,-1,0),-1}, {(-1,0,0),4}, {(0,0,0),-\
4}, {(-1,1,0),-1}, {(1,1,0),1};

Txz5 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,-1,0),-\
2}, {(0,-1,0),4}, {(1,-1,0),-2}, {(-1,-1,1),1}, {(0,-1,1),-2}, {(1\
,-1,1),1};

Txz6 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-1}, {(0,0,-1),1\
}, {(-1,-1,0),-2}, {(0,-1,0),2}, {(-1,0,0),2}, {(0,0,0),-2}, {(-1,\
-1,1),1}, {(0,-1,1),-1}, {(-1,0,1),-1}, {(0,0,1),1};

Txz7 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-\
3}, {(0,0,-1),4}, {(1,0,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),8}, {(1\
,-1,0),-2}, {(-1,0,0),4}, {(0,0,0),-4}, {(-1,-1,1),2}, {(0,-1,1),-\
2}, {(-1,0,1),-1}, {(1,0,1),1};

Txz8 = {(-1,-1,-1),1}, {(-1,0,-1),-2}, {(-1,1,-1),1}, {(-1,-1,0),-\
```

2}, {(-1,0,0),4}, {(-1,1,0),-2}, {(-1,-1,1),1}, {(-1,0,1),-2}, {(-\
1,1,1),1};

Txz9 = {(-1,-1,-1),4}, {(0,-1,-1),-3}, {(-1,0,-1),-6}, {(0,0,-1),4\
}, {(-1,1,-1),2}, {(0,1,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),4}, {(-\
1,0,0),8}, {(0,0,0),-4}, {(-1,1,0),-2}, {(-1,-1,1),2}, {(0,-1,1),-\
1}, {(-1,0,1),-2}, {(0,1,1),1};

Txz10 = {(-1,-1,-1),10}, {(0,-1,-1),-12}, {(1,-1,-1),3}, {(-1,0,-1\
),-12}, {(0,0,-1),12}, {(1,0,-1),-2}, {(-1,1,-1),3}, {(0,1,-1),-2}\
, {(-1,-1,0),-12}, {(0,-1,0),12}, {(1,-1,0),-2}, {(-1,0,0),12}, {(\
0,0,0),-8}, {(-1,1,0),-2}, {(-1,-1,1),3}, {(0,-1,1),-2}, {(-1,0,1)\
,-2}, {(1,1,1),1};

Dy Dz d2u/dydz = Tfyz +

kyz1 Tyz1 + kyz2 Tyz2 + kyz3 Tyz3 + kyz4 Tyz4 + kyz5 Tyz5 +

kyz6 Tyz6 + kyz7 Tyz7 + kyz8 Tyz8 + kyz9 Tyz9 + kyz10 Tyz10,

where

Tfyz = {(-1,-1,-1),-1}, {(0,-1,-1),1}, {(-1,0,-1),3/2}, {(0,0,-1),\
-1}, {(-1,1,-1),-1/2}, {(-1,-1,0),3/2}, {(0,-1,0),-1}, {(-1,0,0),-\
2}, {(0,0,0),1}, {(-1,1,0),1/2}, {(-1,-1,1),-1/2}, {(-1,0,1),1/2};

Tyz1 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-\
2}, {(0,0,-1),4}, {(1,0,-1),-2}, {(-1,1,-1),1}, {(0,1,-1),-2}, {(1\
,1,-1),1};

Tyz2 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-\
1}, {(0,0,-1),2}, {(1,0,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),2}, {(1\
,-1,0),-1}, {(-1,0,0),1}, {(0,0,0),-2}, {(1,0,0),1};

Tyz3 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-2}, {(0,0,-1),2\
}, {(-1,1,-1),1}, {(0,1,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),1}, {(-\
1,0,0),2}, {(0,0,0),-2}, {(-1,1,0),-1}, {(0,1,0),1};

Tyz4 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-\
6}, {(0,0,-1),8}, {(1,0,-1),-2}, {(-1,1,-1),2}, {(0,1,-1),-2}, {(-\

```
1,-1,0),-3}, {(0,-1,0),4}, {(1,-1,0),-1}, {(-1,0,0),4}, {(0,0,0),-\
4}, {(-1,1,0),-1}, {(1,1,0),1};

Tyz5 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,-1,0),-\
2}, {(0,-1,0),4}, {(1,-1,0),-2}, {(-1,-1,1),1}, {(0,-1,1),-2}, {(1\
,-1,1),1};

Tyz6 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-1}, {(0,0,-1),1\
}, {(-1,-1,0),-2}, {(0,-1,0),2}, {(-1,0,0),2}, {(0,0,0),-2}, {(-1,\
-1,1),1}, {(0,-1,1),-1}, {(-1,0,1),-1}, {(0,0,1),1};

Tyz7 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-\
3}, {(0,0,-1),4}, {(1,0,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),8}, {(1\
,-1,0),-2}, {(-1,0,0),4}, {(0,0,0),-4}, {(-1,-1,1),2}, {(0,-1,1),-\
2}, {(-1,0,1),-1}, {(1,0,1),1};

Tyz8 = {(-1,-1,-1),1}, {(-1,0,-1),-2}, {(-1,1,-1),1}, {(-1,-1,0),-\
2}, {(-1,0,0),4}, {(-1,1,0),-2}, {(-1,-1,1),1}, {(-1,0,1),-2}, {(-\
1,1,1),1};

Tyz9 = {(-1,-1,-1),4}, {(0,-1,-1),-3}, {(-1,0,-1),-6}, {(0,0,-1),4\
}, {(-1,1,-1),2}, {(0,1,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),4}, {(-\
1,0,0),8}, {(0,0,0),-4}, {(-1,1,0),-2}, {(-1,-1,1),2}, {(0,-1,1),-\
1}, {(-1,0,1),-2}, {(0,1,1),1};

Tyz10 = {(-1,-1,-1),10}, {(0,-1,-1),-12}, {(1,-1,-1),3}, {(-1,0,-1\
),-12}, {(0,0,-1),12}, {(1,0,-1),-2}, {(-1,1,-1),3}, {(0,1,-1),-2}\
, {(-1,-1,0),-12}, {(0,-1,0),12}, {(1,-1,0),-2}, {(-1,0,0),12}, {(\
0,0,0),-8}, {(-1,1,0),-2}, {(-1,-1,1),3}, {(0,-1,1),-2}, {(-1,0,1)\
,-2}, {(1,1,1),1};

                    Dz Dz d2u/dzdz = Tfzz +

kzz1 Tzz1 + kzz2 Tzz2 + kzz3 Tzz3 + kzz4 Tzz4 + kzz5 Tzz5 +

   kzz6 Tzz6 + kzz7 Tzz7 + kzz8 Tzz8 + kzz9 Tzz9 + kzz10 Tzz10,

    where

Tfzz = {(-1,-1,-1),-1}, {(0,-1,-1),1}, {(-1,0,-1),1}, {(-1,-1,0),2\
```

```
}, {(0,-1,0),-2}, {(-1,0,0),-2}, {(-1,-1,1),-1}, {(0,-1,1),1}, {(-\
1,0,1),1};

Tzz1 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-\
2}, {(0,0,-1),4}, {(1,0,-1),-2}, {(-1,1,-1),1}, {(0,1,-1),-2}, {(1\
,1,-1),1};

Tzz2 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,0,-1),-\
1}, {(0,0,-1),2}, {(1,0,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),2}, {(1\
,-1,0),-1}, {(-1,0,0),1}, {(0,0,0),-2}, {(1,0,0),1};

Tzz3 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-2}, {(0,0,-1),2\
}, {(-1,1,-1),1}, {(0,1,-1),-1}, {(-1,-1,0),-1}, {(0,-1,0),1}, {(-\
1,0,0),2}, {(0,0,0),-2}, {(-1,1,0),-1}, {(0,1,0),1};

Tzz4 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-\
6}, {(0,0,-1),8}, {(1,0,-1),-2}, {(-1,1,-1),2}, {(0,1,-1),-2}, {(-\
1,-1,0),-3}, {(0,-1,0),4}, {(1,-1,0),-1}, {(-1,0,0),4}, {(0,0,0),-\
4}, {(-1,1,0),-1}, {(1,1,0),1};

Tzz5 = {(-1,-1,-1),1}, {(0,-1,-1),-2}, {(1,-1,-1),1}, {(-1,-1,0),-\
2}, {(0,-1,0),4}, {(1,-1,0),-2}, {(-1,-1,1),1}, {(0,-1,1),-2}, {(1\
,-1,1),1};

Tzz6 = {(-1,-1,-1),1}, {(0,-1,-1),-1}, {(-1,0,-1),-1}, {(0,0,-1),1\
}, {(-1,-1,0),-2}, {(0,-1,0),2}, {(-1,0,0),2}, {(0,0,0),-2}, {(-1,\
-1,1),1}, {(0,-1,1),-1}, {(-1,0,1),-1}, {(0,0,1),1};

Tzz7 = {(-1,-1,-1),4}, {(0,-1,-1),-6}, {(1,-1,-1),2}, {(-1,0,-1),-\
3}, {(0,0,-1),4}, {(1,0,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),8}, {(1\
,-1,0),-2}, {(-1,0,0),4}, {(0,0,0),-4}, {(-1,-1,1),2}, {(0,-1,1),-\
2}, {(-1,0,1),-1}, {(1,0,1),1};

Tzz8 = {(-1,-1,-1),1}, {(-1,0,-1),-2}, {(-1,1,-1),1}, {(-1,-1,0),-\
2}, {(-1,0,0),4}, {(-1,1,0),-2}, {(-1,-1,1),1}, {(-1,0,1),-2}, {(-\
1,1,1),1};

Tzz9 = {(-1,-1,-1),4}, {(0,-1,-1),-3}, {(-1,0,-1),-6}, {(0,0,-1),4\
}, {(-1,1,-1),2}, {(0,1,-1),-1}, {(-1,-1,0),-6}, {(0,-1,0),4}, {(-\
1,0,0),8}, {(0,0,0),-4}, {(-1,1,0),-2}, {(-1,-1,1),2}, {(0,-1,1),-\
```

```
1}, {(-1,0,1),-2}, {(0,1,1),1};

Tzz10 = {(-1,-1,-1),10}, {(0,-1,-1),-12}, {(1,-1,-1),3}, {(-1,0,-1\
),-12}, {(0,0,-1),12}, {(1,0,-1),-2}, {(-1,1,-1),3}, {(0,1,-1),-2}\
, {(-1,-1,0),-12}, {(0,-1,0),12}, {(1,-1,0),-2}, {(-1,0,0),12}, {(\
0,0,0),-8}, {(-1,1,0),-2}, {(-1,-1,1),3}, {(0,-1,1),-2}, {(-1,0,1)\
,-2}, {(1,1,1),1};
```